UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SECURITY METRICS TO EVALUATE QUALITY OF PROTECTION

VERSÃO PUBLICA

Alberto Manuel Giroto Bruno

MESTRADO EM SEGURANÇA INFORMÁTICA

Abril 2011

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

SECURITY METRICS TO EVALUATE QUALITY OF PROTECTION

Alberto Manuel Giroto Bruno

Orientador
Rahul Telang
Co-Orientador
Marcelo Pasin

MESTRADO EM SEGURANÇA INFORMÁTICA

Abril 2011

Resumo

As organizações precisam estar conscientes de qual o grau de protecção contra ameaças à segurança da sua informação.

A fim de establecer prioridades nos investimentos, as organizações precisam conhecer o "estado da arte" da sua segurança, para definir que sistemas precisam, mais cedo, de mais atenção. Isso só é possível com números que permitam identificar o nível de segurança dum sistema de informação e compará-lo com os outros. Para atingir este objectivo, as organizações devem definir quais são os factos relevantes que devem ser medidos, e como extrair informações significativas a partir dessas medidas.

As métricas de segurança ajudam as organizações a compreender a Qualidade de Protecção (QoP - Quality of Protection) dos seus sistemas de informação. Estas são uma valiosa ferramenta para diagnosticar problemas. No entanto, devem ser usadas com precaução, uma vez que, por exemplo, valores idênticos podem, em contextos diferentes, não ter o mesmo significado. O que exige sentido critico, ao técnico que analisa tais metricas, para perceber o seu real significado. As métricas de segurança devem ser usadas da mesma forma que um médico usa os resultados das análises de sangue, ou da pressão arterial, não como um resultado *"per se"*, mas como um indicador de um possível problema.

Este projecto tem como objectivo a implementação de um conjunto de métricas de segurança, para avaliar quão vulneráveis estão os sistemas de informação duma organização e qual é o nível de ameaça que enfrentam. Pretende-se ainda combinar essas métricas para determinar a sua QoP.

Para atingir estes objectivos, vamos identificar que métricas são pertinentes para cada tipo de componente do sistema de informação, vamos definir um modelo para calcular as métricas recolhidas e obter a informação sobre o risco de segurança do sistema, e finalmente vamos submeter a nossa proposta a um sistema real para verificar a aplicabilidade prática deste trabalho.

Palavras-chave: Segurança, Metricas, Qualidade de Protecção, QoP, Vulnerabilidade, Ameaça, Risco

Abstract

Organizations need be aware of how protected they are against information security threats.

In order to prioritize their investments, organizations need to know the "state of the art" about their security, to determine which system needs closer attention, sooner. This is only possible with numbers that allow define system's security level, and compare it with others. In order to achieve this goal, organizations must define which are the relevant facts that should be measured, and how to extract meaningful information from those measures.

Security metrics help organizations to understand the QoP (Quality of Protection) of their information systems. Security metrics are a valuable tool for diagnosing problems. Nevertheless, it should be used with caution, given that, for example, identical values may, in different contexts, not have the same meaning, which demands critical judgement by the technician, which analyzes such metrics, to realize their true meaning. Security metrics should be used the same way a physician uses blood test results, or blood pressure monitoring, not as an outcome "per se" but as an indicator of a possible problem.

This project aims to propose the implementation of a set of security metrics, to evaluate how vulnerable are information systems, and what is the threat level they face. We also want to combine these metrics to determine their QoP.

To achieve our goals, we will identify which metrics are relevant for each type of information system component, we will define a model to compute the metrics collected and derive the information system security risk, and finally we will apply our proposal to a real system to verify the practical applicability of this work.

Keywords: Security, Metrics, Quality of Protection, QoP, Vulnerability, Threat, Risk

Acknowledgments

I would like to thank my advisor Professor Rahul Telang for his valuable advices, and guidance. Professor Marcelo Pasin for his availability.

I also would like to thank my collegues and friends at Portugal Telecom who helped me to collect data from systems, and provided me technical support, a special thank to José Aser, Ricardo Oliveira and Luis Sousa.

Finnaly I would like to thank my sister Marta for her availability, and support in the statistical area.

Lisboa, April 2011

Dedicated to Sonia, Sofia and Ricardo

for all their love, affection and patience

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Information security is increasingly a topic on agenda. Every day we read news about system intrusion, data breaches, identity theft, denial of service, viruses, worms, etc. Some consequences of such events are reputation damage, or losses either direct (e.g., due to productivity decrease, and costs to fix the damage) or indirect (e.g., due to liability, and market-share reduction). To know how secure a system is, or how protected it is against these threats is a growing concern for organizations. Therefore organizations wonder if they are safe, or which is their degree of exposure.

A wise mind knows that there is no such thing as perfect security in information systems. Thus the goal must be to achieve a security level close to perfect security, but without incur in higher costs than the value of the protected object.

In order to decide where to invest and how to prioritize investments, organizations need to know how well protected each system is, which systems have higher risks, and how much should be invested to achieve the desired protection level. This is possible only with numbers that allow setting the security level of a system and with comparison of the level of protection of each system. To obtain relevant information, organizations must to define what should be measured, i.e., which are the facts that have impact in security, and how to extract meaningful numbers from that facts.

The measurement field in information security is not, however, as developed as, for instance, the field of QoS (Quality of Service). Which security metrics should be used, what they mean, how they relate to each other, and how to extract coherent numbers from them is far from being a consensual matter. *"Even the term "security metrics" is often ambiguous and confusing in many contexts of discussion in information security*" [1]. Thus in spite of the growing attention that IT (Information Technology) professionals have devoted to the QoP subject, and the general consensus about the fundamental role metrics play in better knowing security issues; it still needs deeper study, and it has not yet gained broad agreement.

Mainly due to subject immaturity, but also due to the information sensitivity of the measured object in this field, it is often difficult to collect data to calculate metrics. Also as stated in [2] "*Practical challenges range from the lack of common definitions for terms and metrics to determining how to share information meaningfully. The vocabulary of information security is fraught with imprecision and overlapping meanings. Fundamental concepts and words, such as incident, attack, threat, risk, and vulnerability, mean different*

things to different people. Basic metrics for counting events are difficult to collect because the terms are unclear."

Still, we believe that the use of security metrics is the way to achieve a better understanding about the QoP of organization's information systems. Security metrics can help in diagnosing the problems, and should be seen the same way the physician sees prescriptions for blood tests, or level of blood pressure. Sometimes the same results may have different meanings; it is the human's educated mind that has to extract information from such results.

In this context, this project main goal is to identify how well protected is an asset or an information system "per se" (we consider a system as a group of assets working together for the same purpose), based on security metrics collected from the systems them self, from defence mechanisms and from publicly available data. Also, we totally agree with the sentence: *"The scientific research world has always split into those two broad camps. The theorists rely on the experimentalists to gather the data by which competing theories are tested, and the experimentalists rely on the theorists to show how to extract order, structure and knowledge out of the mass of data."*[1] Thus we believe that just collect data is not enough, and that, some usefull information should be extracted from such data. Therefore, we are ambitious about doing both, so a model to compute the collected information and derive the system's QoP, will be proposed.

The model aims to identify the risk to which an information system is subject. This model is based on the equation $Risk = Threat \times Vulnerability$ and with this project we want to:

- suggest several metrics that we consider representative of the information we want to get (we intend that this set of metrics be automatically or easily collected, and we will not use metrics derived from questionnaires or from expert opinion);
- define a high level process of information gathering, suggesting how and where this metrics can be collected;
- define a model that allows to organize and to combine each metric in a coherent set, and which will allow to derive a risk index for an information system;
- collect the defined metrics from a real live system in order to prove the practical applicability of our model.

More than a closed and final work, we seek that current work be a starting point for organizations begin collecting and correlating security data, producing indicators that will improve knowledge of information security, allowing to check their evaluation over time. New metrics may be (and should be) added to our model either due to technological evolution (which, for instance, will lead to new threats, or will made possible new data gathering), or because new systems or new components are added to information systems (different systems/components may have different security metrics depending of factors like architecture, hardware and software used, among others).

This work also intends to be an add-on to Pulso[2] application, so its development will take into consideration functionalities already implemented by Pulso.

[1] John Leach, securitymetrics.org mailing list message, "Modelers v measurers (was: Risk metrics)," January 31, 2006. [2]
[2] System developed at Portugal Telecom to assess their systems' QoS.

Note: the present work has a confidential version. Some contents were removed from this public version, namely: confidential data from several tables, confidential data from graphics, and a part of Section 5.9. In tables that had some correlation with other tables, or graphics , we replaced the data by variables, otherwise the data was simply removed.

Chapter 2

Related Work

2.1 OCTAVE – Allegro

OCTAVE-Allegro [3] intends to be a more agile version of OCTAVE [4], evaluating organization's operational risk environment, focusing the assessment in information assets based on their journey thru organization assets, seeing all other assets as mere containers where information assets are stored, transported or processed. *"A container can be a person (since people can store information as knowledge, transport information by communicating, or process information by thinking and acting), an object (e.g., a piece of paper), or a technology (e.g., a database). Thus, threats to information assets are identified and examined through the consideration of where they live, which effectively limits the number and types of assets brought into the process"* [3].

OCTAVE-Allegro defines eight process steps (the output of each step will be used as input to the next one) grouped in four phases:

1. Establish Drivers

   defines risk measurement criteria, identifying the organizational drivers and the most sensitive risk areas for organization mission and objectives;

2. Profile Assets

   creates a consistent description of information assets and their security requirements and identifies the containers where they are processed, stored, and transported, whether internal or external to organization;

3. Identify Threats

   real-world scenarios are surveyed, identifying areas of concern and their corresponding undesirable outcomes, these areas of concern are then expanded to threat scenarios that further detail the properties of a threat;

4. Identify and Mitigate Risks

this phase ensures that consequence of risk are captured, then a quantitative measure of the impact a threat may cause in the organization is computed, and a relative risk score is derived. Finally risks are prioritized based on their relative scores, and their mitigation strategy is defined.

OCTAVE-Allegro is an outstanding and well-structured process to think about the security risks a organization faces. Nevertheless, implementing such a process will be heavily time/resource consuming and in consequence most organizations will not be able to deploy it. Therefore, we intend to present a light weight process for measure QoP. Our process will implement a different perspective, instead of defining how well protected is an information asset all over its containers, will define for each information system (which will be a set of technological containers) how well protected are all the information assets it contains. Our method will allow a gradual approach to security measurements instead of all at once.

2.2 ISO/IEC 27004 Information Technology – Security Techniques – Information Security Management – Measurement

It is an international information security standard developed by ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission).

It is devoted to development and use of measurement in order to determine the effectiveness of an ISMS (Implemented Information Security Management System) as specified in ISO/IEC 27001.

This standard provides an information security measurement overview, describes the activities involved in a program of measures and measurement, and defines its objectives and success factors.

Consists of the following sections:

1. Management responsibilities

   defines the management responsibilities in establishing the program involving relevant stakeholders in measurement activities, verifying measurement results, and improve the ISMS based on gathered results;

2. Measures and measurement development

   provides guidance on how to develop measures and measurement for the purpose of assessing the effectiveness of ISMS; defining measurement scope, information needs, objects of measurement, development process, data collection process and analysis, implementation, and documentation;

3. Measurement operation

   establishes how security measurement activities such as: data collection, data storing, data verification, and integration on the operation of ISMS should be conducted in order to provide accurate information about effectiveness of ISMS;

4. Data analysis and measurement results reporting

   defines how gathered data should be analyzed, how to develop measurement results and to communicate results to relevant stakeholders;

5. Information Security Measurement Programme Evaluation and Improvement

   establishes the need for regularly evaluate the measurement programme effectiveness, guarantying that: the program produces relevant results in accordance with information needs, is executed as planned, addresses changes in ISMS, and implements improvements.

This standard deeply details the measurement process and is a good guide, on how to implement a measurement programme, and in detailing the activities involved. For instance, it describes in detail how to collect, aggregate, derive measure, and produce indicators. Nevertheless, it is short on information about which concrete measurements can be collected (e.g., which attributes may be relevant for some object of measurement) in order to identify the ISMS effectiveness.

2.3 Relative Attack Surface Measures

The work presented in [5] proposes a metric which allow to compare the security of two systems by measuring and comparing their attack surface, *"rather than attempt to measure the security of a system in absolute terms with respect to a yardstick, a more useful approach is to measure its "relative" security. We use "relative" in the following sense: Given System A, we compare its security relative to System B, and we do this comparison with respect to a given number of yardsticks, which we call dimensions. So rather than say "System A is secure" or "System A has a measured security number N" we say "System A is more secure than System B with respect to a fixed set of dimensions""* [5].

It starts by defining a set of attack vectors, which then are categorized into different abstract dimensions defining the system's attack surface.

The defined dimensions are: process targets, data targets, process enablers and channels, where targets can be constrained by access rights. The measure of the attack surface is then a function of the contributions of each dimension to the attack surface and, in turn each dimension is a function of the attack vectors to which a weight is attributed in function of its importance to the dimension it belongs.

This method can only be applied to systems that have the same operating environment and the same threat model. It is useful only to compare different versions of the same system or the same system with different configurations.

Chapter 3

Computation Model

The aim of this study is to suggest a practical approach to security metrics usage. In this chapter, we intend to define a simple model, easy to deploy, and adaptable, which should help organizations to give a first step toward defining, extracting, collecting and correlating useful information about the QoP of their information systems.

3.1 Intended Characteristics

We intend that our model is easily implementable and flexible enough to allow the introduction of new metrics to comport and to be adaptable to:

- constant information technology evolution, which leads to deployment of new products, evolution of old ones and to the consequent emergence of new threats and vulnerabilities from time to time;
- creative and innovative nature of attacks, which will probably lead to finding new kinds of vulnerabilities;
- different needs from different organizations, which will not be able/want to collect the same metrics (a metric that is important for an organization may not be important in other contexts, additionally, an organization can deploy new defences and most be able to understand how these new defences improved his security);
- organization expertise growing, after using security metris for a while, organizations will likely find out that some of the metrics initially collected, do not give the expected information and will also realize that others may be collected and tried in order to improve the QoP indicator;
- different attack surface of assets, which influence the possible kind of attacks it can suffer; consequently metrics that are meaningful for some kind of asset or component may be meaningless for another (different metrics can be defined for each kind of asset, and would be impossible in the context of this work to define all sets of metrics for each kind of asset).

Therefore, a rigid model can become outdated sooner than it is deployed, so only an adaptable model which is able to comport, and extract useful information from new metrics without jeopardize the old ones, will succeed.

It is also an objective of this work to produce information that can be easily understood by non-technical staff. Looking at the results produced by the model should be clear for non-technical people which is the QoP presented by an information system.

This model is focused in define/compute security metrics to measure QoP in scenarios of unauthorized access from malicious entities, either by failure of proper controls, or by vulnerability exploitation. However we believe that with an appropriate set of metrics and data it can be extended to measure QoP in unintentional failures/attacks scenarios.

3.2 Terms Definition

Information System designates a collection of assets, each one with specific roles, which collaborate in order to provide a set of related business functions to the organization.

Asset a single equipment composed of hardware and software, which has a specific role in an information system, for instance, application servers, or database servers are assets.

Component element that is part of an asset, for instance operating system, or database management system. Each component will have a specific set of group metrics.

Metric Group is a set of related metrics, which can produce an indicator about some aspect of system security. For instance we will define a set of metrics that are related with patching activity and we will group all of them in a group called Patching.

Metric is a number that measures some security related issue, or a logical value indicating the presence or absence of a security mechanism. It can be a cardinal number, a percentage, an average, or a boolean value.

Sub-metric represnts a sub-part of a metric, it is used when a metric can be specialized in several parts. For instance, if one have information allowing to define several impact levels (like low, medium, or high) one can create sub-metircs according to it, each part will have a different contribution to the overall metric.

3.3 Model's Rational

The main block of our model is based on the well known equation $Risk = Threat \times Vulnerability$. According to [6], *"The correct measure of how secure a system is depends as much on the number and severity of the flaws of the system (vulnerability) as on the potential of the attacks it may be subjected to (threat). This measure is called risk, (...)* ***Risk*** *- combined measure of the level of threat to which a computing*

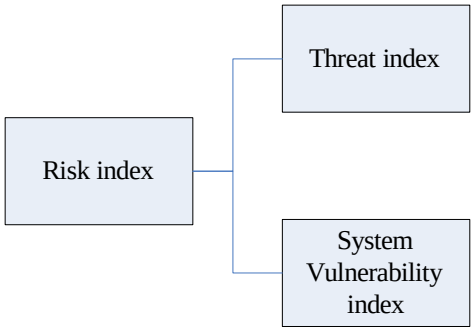


Figure 3.1: Risk Index Graph

or communication system is exposed, and of the degree of vulnerability it possesses". Where Vulnerability is a *"non-malicious fault or weakness in a computing or communication system that can be exploited with malicious intention"* and threat is the *"potential of attack on computing or communication systems"*.

Therefore this is our starting point and our goal is to define and to gather information that will allow to calculate the level of threat and the level of vulnerability of a system. Then based on these two parameters we will compute the risk of the system as shown in Figure 3.1. It becomes obvious that the higher the risk the lower the QoP (which is our final objective). To calculate threat and vulnerability indexes, our model will combine several sets of metrics collected from different data sources.

Firstly, we use a graph to represent the model. In the graph, each leaf will have a base metric; each parent node will have a value computed from a set of base metrics, collected from several data sources or from their child nodes; and each edge will have a weight representing the relative importance of the node to its parent node (for instance, in an information system that uses a central authentication server, the weight of this asset should probably be higher than the weight of other assets, due to its importance in keeping the information system secure). This mechanism will allow to define distinct levels of severity for each metric according to its importance to the overall vulnerability/threat of the information system.

This graph representation is inspired in Pulso's QoS calculation model [7], and the graph structure used here is mainly imported from that model. Even though, it is not a request from PT (Portugal Telecom), which proposed the thesis theme, it is our understanding that the similarity among the two models it is natural and desirable. On one hand it is well suited to our calculation needs, and on the other hand it will be beneficial for PT since it will allow reusing some pieces already in place, and will also facilitate model understanding for non-technical people which is already familiarized with the QoS model.

Notwithstanding we do not present (in the context of this work) an implementation of our model, it is our understanding that even if the model has to be developed from scratch, it will be a straight forward process. Thus any organization besides PT can take advantage of it with little effort.

3.4 Threshold Definition

In order to extract some meaning and order from each metric and to avoid that metrics with higher nominal space of results to have, just because of that fact, a superior weight; it is important that all metrics are transformed in a common set of categories independently of their different natural space of results. For

instance if we are measuring the number of firewall blocked connections, what does means a value of 1500? Is it high or is it low? And how do we mix it with a metric with a different space of results like, for instance, the percentage of spam in received mail. If we do not adjust the scales it will be like comparing apples with oranges. Therefore, to simplify model computations, each and every metric must be converted to a common scale. In our universe, one possible approach is to define a set of categories which will define a level of impact in terms of security. For instance we can define a Likert scale with the following five categories:

- 0 - no security impact;
- 1 - minimal security impact;
- 2 - medium security impact;
- 3 - high security impact;
- 4 - top security impact.

Then each metric's value must be converted to 0, 1, 2, 3 or 4 according to the impact it will have in terms of vulnerability/threat.

This process raises, however, another question: how to convert metrics' values to one of these categories? It is our understanding that several methods can be applied, and different metrics may employ different methods, depending on the metric characteristics (space of results, dispersion, sample size, etc). One method may be more adequate than other. We can devise four main methods to achieve this goal:

1. Causation

   Causation can be used if we know, supported by evidence, that a certain value of the metric will cause a certain effect; i.e., that the same value of the metric will cause always the same impact. In security field, this method seems hard to apply since things are not completely black or white and usually depends of the conjugation of several factors.

2. Expert knowledge

   This method is based on the empirical knowledge of individuals that worked on the area, which is being object of study, for a long period of time. These individuals, by experience, know that when a metric reaches a certain value probably an impact of certain level will happen. Combining the knowledge of a high number of experts will certainly improve classification quality.

3. Statistical analysis

   If we have samples that are representative of all metrics' values and that are big enough we can apply statistical analysis to that sample, in order to help us determining which values belong to which category. The rational behind this method is that an attack has a high impact only if the organization is not prepared to deal with that attack. Also an organization to survive has to be well adapted to the conditions of the environment where it belongs, so it is expectable that an organization is prepared to deal with the usual attack level in its environment, i.e., it has an adequate number of resources to face it. Therefore if we can extract from our sample which is the usual attack level we can state that every thing above will represent a high impact and every thing bellow will represent a low impact.

4. Comparing with a normal pattern

   If we have access, for instance, to a public reliable source, and if we can extract a definition of what is normal then we can use this data to help setting the categories of a metric.

3.5 Weight Definition

As stated before, different metrics will have different relevance in defining a global index, so our model addresses this issue by assigning a weight to each metric; this weight defines the relative importance of a metric in relation to its brothers (see Figure 3.2).

In the first stages of model implementation, it is unlikely that one have enough information correlating security incidents and metrics, in order, to allow us defining accurate weights for each metric or even for each metric's group. So as a first approach one will have to rely on experts' opinion to obtain weights. Also, hardly one will find two experts that assign exactly the same weights to the same metric, but this is an insuperable constrain. Therefore the best way to improve the quality of weights (in the first stages) is to increase number of inquired experts and combine their opinions.

One can also perform sensitivity analysis to determine the influence each metric and its respective weight have in the overall index, in order to adjust weights accordingly.

When the model is in production for some time and some experience and causation information exists, then the weights can be fine tuned in order to improve the model results.

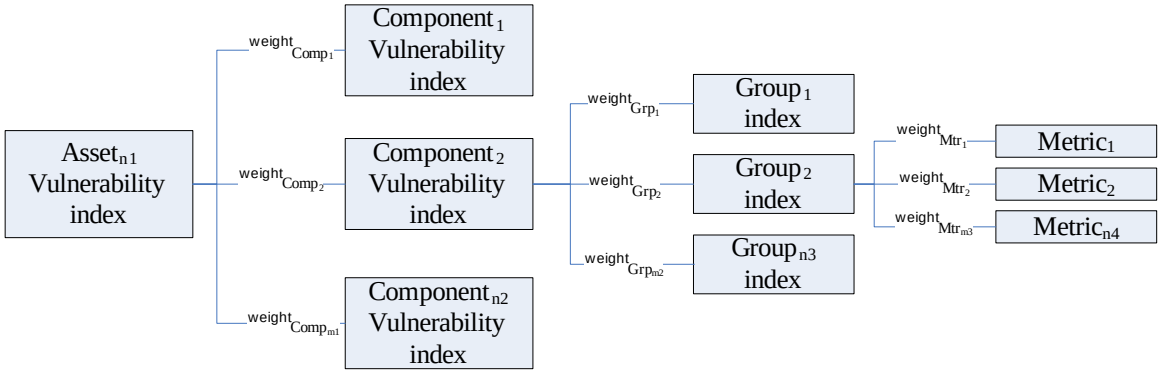


Figure 3.2: Graph Weights

3.6 Vulnerability Index Calculation

Figure 3.3 represents the vulnerability branch of our model, the tree has four levels representing: assets (e.g., web server, database server), components (e.g., database management system, operating system), security groups (patching, access control), and base metrics.

The leaves of our tree have the base metrics (in some particular cases, we can have sub-metrics, but they will end up producing a metric, so for sake of simplicity, we will not consider them here) collected from the components of each asset. Each metric is computed in order to match an interval threshold.

These metrics are organized in groups, each representing some aspect related to security (e.g., patching, or access control); for each group, we will compute an index based on the weights and thresholds of its leaves.

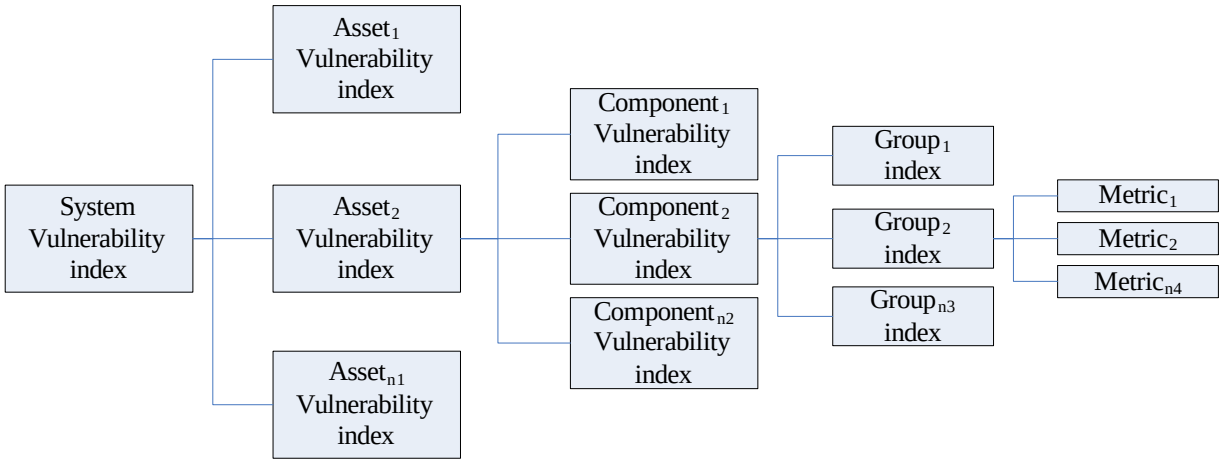


Figure 3.3: Vulnerability Branch Graph

The calculation of vulnerability index is based on the idea that each metric will contribute with its share to the overall vulnerability of the information system. Thus to calculate the parent node index, we use a weighted mean of its child nodes.

$$index\,(Parent) = \frac{\sum_{i=1}^{n} (weight\,(Child_i) \times index\,(Child_i))}{\sum_{i=1}^{n} weight\,(Child_i)}, \tag{3.1}$$

where n is the number of child nodes, and $weight$ is the function that gives the weight value of $Child_i$ node.

3.7 Threat Index

Our first approach to define threat index was to calculate it in a similar way as the vulnerability index, i.e., using a bottom up method, calculating a threat index for each system component and use the obtained values to calculate the asset threat index and do likewise for the upper level indexes. However the data that allow us to define the threat index is usually produced to fulfil macroscopic level necessities. Therefore the task of defining the threat index tends to follow a top down approach. Thus, we have a set of metrics collected at the organization level that will be shared by all organization systems. Depending on the availability of data, some metrics may be collected at a lower level, e.g., at data centre level where typically systems share the same perimeter defences, so the same metric's values will be assign to systems sharing the same defences. This means in practice that the same threat index will be shared by several systems.

One can argue that this approach it is not correct, because different system characteristics like sensitivity of stored information, may pose different threats to information systems, for instance, a system with confidential information about a new product will be more interesting to an attacker than a system with public data, and this way the first will face a higher threat. We agree with this argument and our model is prepared to deal with that, since new metrics can be added. Therefore the index specialization will depend only of the ability to collect more specialized metrics. In the context of this work and as a first approach, it is our understanding

that it is a reasonable assumption to say that systems that share a common location and a common set of defences will share an identical level of threat, and thus define a broader index that will be shared by several systems.

The Figure 3.4 represents the threat branch of our model. In this case, the tree has only three levels representing: level at which the metric was collected (e.g., organization, data centre, room, system), threat groups (e.g., spam, viruses, incidents), and base metrics.

The weighted mean method will, also be used, to calculate the threat index, once it is our understanding that the overall threat will be influenced by each one of the identified threats. So to calculate each index we will use Equation (3.1), described above.

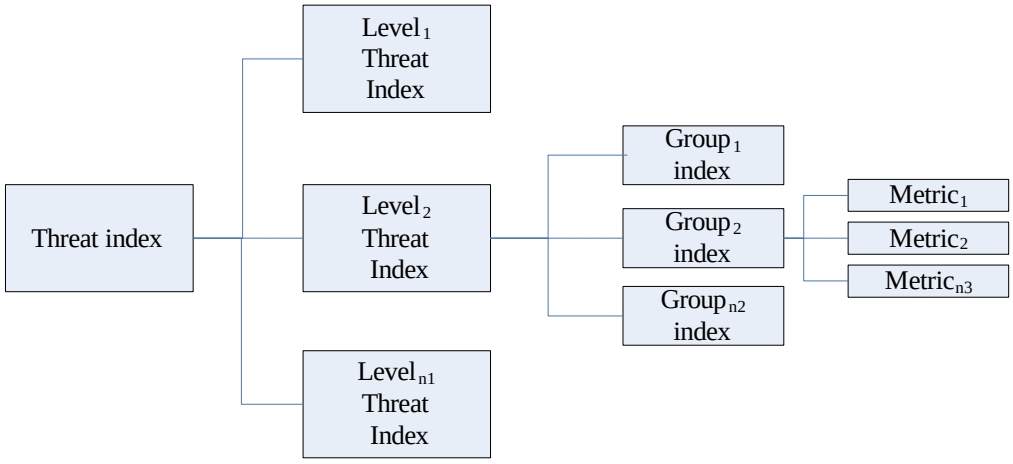


Figure 3.4: Threat Branch Graph

3.8 Risk Index Calculation

The risk index will be a simple product between vulnerability index and threat index, both contributing in equal parts for final outcome. Since threat and vulnerability indexes will have values between 0 and 4, the risk index will have values between 0 and 16, where 0 means that the system has no risk at all (either because it has no vulnerabilities or because it is not exposed to any threat), and 16 means top risk (because it is exposed to the maximum threat level and it is extremely vulnerable).

Chapter 4

Metrics

In this chapter, we will propose a set of metrics that we consider to have high importance in defining the level of vulnerability/threat an information system possesses. The list of metrics presented do not intend to be exhaustive nor suitable to all systems, since as stated before that would be an impossible mission in the context of this work. Such a complete set of metrics requires deeper analysis of the system by organization security experts of several areas of knowledge (networking, software, hardware, etc), and may not be the same for all kinds of system, for instance, Windows systems may have different metrics than Unix systems. Also new architectures, assets, components and even attacks will appear from time to time, and these events will force the definition of new metrics to improve index accuracy and completeness. Certainly due to the spread use of identical architectures, the presented metrics will be suitable, at least, as a first QoP approach for most information systems.

In this chapter we will consider two main distinct groups of metrics, the ones that will contribute to vulnerability index and, the ones that will contribute to threat index.

Each metric will be expressed in one of the following forms:

- # – "number of", which will give us absolute values of some measured element;
- % – "percentage of", which will give us the percentage of a measured element related to the total number of elements;
- avg – "average of", which will give us the mean value of a set of measured elements;
- boolean value.

Some times a metric may be available in more than one form, in such situation, only the one that come to be consider most accurate in each case, should be used.

4.1 Vulnerability Metrics

All security options must be ruled by a security policy, which should define what is secure, and which levels of security should be achieved. Therefore, the base set of metrics must contain (but do not need to be limited to it) those that allow identifying gaps between security policy and the information system implementation.

Group	Metric	Component			
		Operating System	App. Server and Web Server	DBMS	Bussiness Applications
Patching	security patches not installed	√	√	√	√
	time taken to install security patches	√	√	√	√
Access Control	weak/default passwords	√	√	√	√
	accounts with unchanged passwords for more than n days	√	√	√	√
	security misconfigurations detected	√	√	√	√
	users with dangerous privileges	√	√	√	√
Anti-virus	anti-virus installed?	√			
	days old of signatures file	√			
	virus detected	√			
Known Vulnerabilities	open ports	√			
	open ports in top 10 list	√			
	vulnerabilities detected from vulnerability scan	√	√	√	√
	unneeded software installed	√	√	√	√
Code Quality	LoC (lines of code)				√

Table 4.1: Components, Groups and Metrics

Each kind of asset can have differents components, i.e., each asset will have only a subset of the components here presented. For instance, in a three tier architecture, usually the DBMS (Database Management System) and the Application Server are installed in different assets, but it is also possible to have them sharing the same asset.

Similarely each component may not embody all groups of metrics, also some metrics (in spite of being defined in one group) may not make sense depending on the combination of component and group. For instance, not all metrics, defined for operating system known vulnerabilities group, will make sense in the context of DBMS known vulnerabilities, like for instance, open ports. Even though, a database can have several listeners, its cardinality will not be very informative and at least in a first approach to security metrics it should not be considered.

Therefore, in this section, first we will present the kind of components usually found in information systems' assets and their usual weak points. Then we will present the metrics we considered in this work, grouped by security affinity. Table 4.1 summarizes, which metrics belong to wich groups, and which groups belong to which component.

In addition, it is worth noting that some metrics will not be easy to gather, and some of them probably will be impossible during the present work, therefore, albeit we refer them in this chapter not all of them will be part of our practical work.

4.1.1 Components

4.1.1.1 Operating System

It is well known that most of the attacks perpetrated against information systems take advantages of weaknesses in operating systems. It is also well known that weaknesses in some operating systems are more prone to be discovered than in others, either because less care was taken in their development and testing

process, or because they are widely deployed and exposed to a larger scrutiny. In addition, the access control mechanisms (password strength, user privileges, access control lists, etc) and their configuration represent an important role in systems security. Therefore, when we talk about security metrics, this component is of central importance to identify the vulnerability index of the system. We have identified four groups of metrics (patching, access control, anti-virus, and known vulnerabilities), which we believe are good indicators of operating system level of vulnerability.

4.1.1.2 Database Management System

Nowadays almost every information system is composed by a DBMS, which plays a role of major importance since it is the repository of the data, which is often the target of the attacks. Besides code vulnerabilities which is common to every piece of software in information systems, DBMS are also subject to sql injection attacks (ranked 2 in [8]), weak passwords, accounts with excessive privileges, improper access control, among others. In this work we focused on three groups of metrics (patching, access control, and known vulnerabilities).

4.1.1.3 Application Server and Web Server

Application servers and web servers play a central role in the widely deployed multi-tier layer architectures, so they are unavoidable when we think about information system security. In [8] we can find several examples of vulnerabilities respecting this type of system assets. These are the components, which usually are in direct contact with the "outside world", this way authentication and authorization mechanisms, and patching are essential to guarantee a good level of security.

4.1.1.4 Business Applications

Like any other piece of software, business applications that compose the system are subject to bugs and vulnerabilities, either when developed in-house or by third parties. The relation between number of code lines and bugs of an application has been long studied and results show that rigorous quality assurance may reduce in approximately ten times the number of bugs and consequently the number of security vulnerabilities. Thus, besides the three groups used in prior components, we also consider code quality an important group, in this component.

4.1.2 Groups

4.1.2.1 Patching

As stated in [2], *"patching is an essential part of keeping systems up-to-date and in a known state. In other words, it is part of an overall portfolio of security controls. The degree to which an organization keeps its infrastructure up to patch indicates the effectiveness of its overall security program. It is likely that organizations that do not patch their systems have highly variable system configurations that render other security controls ineffective."*

Even though, is not wise patching a system as soon as a patch is released, and prior to validate its full impact in the system, it is even less wise do not patch the system or take to long to apply a patch. Patching metrics

can give us information about the sanity of a system and the care taken with its management. Bad patching performances are an indicator of a less secure system.

For this group we considered the following metrics:

- security patches not installed;
- time taken to install security patches.

The first metric, will count the number of operating system missing patches, and the second will measure the time a patch took to be installed. Thus, this group of metrics will give us some insight about how careful it is the system managed in relation to security issues. If a high number of security patches are missing and patch installation takes too long then, with high probability, the system management is not very aware of security issues and the system will tend to be an insecure system.

4.1.2.2 Access Control

Access control plays a fundamental role in keeping information systems secure. It sets who has access to the system, to which objects, and with which privileges. To avoid improper appropriation of information, or system damage the establishment of secure credentials, setting correct access rights and privileges, and the enforcement of security policies are essential tasks to keep systems secure. As stated in [9], *"Access control policies and their enforcement mechanisms address two specific threats. The first is a legitimate user who is trying to gain access to information he is not supposed to have or perform functions he's not authorized for. The second threat is a hacker who has broken into a legitimate user's account."*

For this group we have considered four metrics:

- weak/default passwords;
- accounts with unchanged passwords for more than n days;
- security misconfigurations detected;
- users with dangerous privileges.

The first two metrics of this group, like in previous group, will give us some insights about system management positioning with regard to security. If security policy is enforced in a system then outdated passwords will not be found in the system. Moreover, the number of easily guessable passwords will be near to zero. Besides, this two metrics will also give us some insights about users positioning with regard to security, because, even if the policy is not enforced, the user may proactively change is passwords in a regular way and define secure ones.

The third metric, measures the number of users with access to an object that it should not have access, because that object is not needed for him to execute its work. The fourth metric, counts the number of users with excessive privileges, i.e., the number of users having privileges usually attributed only to "super-user" accounts. Due to business needs, time-to-market pressure, lack of proper tools or their incorrect use, process flaws, among other reasons, often, privileges and access grants have to be set in an had-hoc way, and some times their revocation is forgotten when they are no longer need [10]. Thus, it is highly probable that some of these events appear even in a system with a tight management. Nevertheless, high numbers in these metrics are strong indicators of a sloppy management, or an unstable system, which configures a vulnerable system.

4.1.2.3 Known Vulnerabilities

This set of metrics addresses issues related to well known sources of vulnerabilities. It will give us indication about cleanse quality of the system. A system that is not free of impurities and have, for instance, more open ports, or more software pieces that it needs to perform its function is a system that increases its vulnerability index, since it opens unnecessary doors which can be maliciously exploited.

For this group we considered the following metrics:

- open ports;
- open ports in top 10 list;
- vulnerabilities detected from vulnerability scan;
- unneeded software installed.

The first two metrics are the most elementary ones; since a port is a point of entrance of possible attacks then the probability of a successful attack increases for each open port. If the open ports are the ones that are most attacked, then the probability of a successful attack his even higher. Attackers use automatic software to identify system vulnerabilities and will prepare their attacks based on the information supplied by that kind of software, therefore the number of vulnerabilities report by vulnerability scanners are good indicators of security, even though, they can be prone to false positives. Also unneeded software, i.e., software installed but that it is not necessary for system to perform its functions, is an unnecessary source of vulnerabilities. But, even though, this is an interesting metric it is highly dependent of the server functions and will be hard to collect automatically, also it will have slight variations from one period to another (after a sanitization process it will probably produce values that will be classified as no security impact category) so instead of being part of a metric system it probably should be part of a regular systems' audit and consequent sanitization.

4.1.2.4 Anti-virus

This group will help us to understand how vulnerable a system is to viruses' attacks. In [9], Thomas Parenty states *"Viruses are probably the most visible information security problem computer users experience personally. Spreading quickly and quietly, they can cripple a company's email, corrupt or delete important computer files, or halt worker productivity until they're eliminated."* To make matter worst, it's almost impossible to keep viruses one hundred percent apart from organization assets. Nevertheless, some measures may be deployed to mitigate the number of successful viruses' attacks and their impact. Therefore, the need to know the degree of vulnerability that a system has in relation to virus is not at all negligible.

For this group we have identified three metrics:

- anti-virus installed?;
- days old of signatures file;
- virus detected.

The first two metrics will give base information about anti-virus policy implementation while the third will help to understand the effectiveness of the countermeasures installed.

A more exact way to identify how outdated is a signature file, would be verifying how many signature files have been released after the one installed. But that metric is harder to obtain because it forces us gathering and comparing data from two sources, therefore, since anti-virus companies release new signatures on an almost daily basis, counting the days since the signatura file was installed will give us an identical value with less effort.

4.1.2.5 Code Quality

Vulnerabilities in code can exist either unintentionally, or intentionally (in some specific cases), both representing exploitability opportunities. Code quality can be improved by adopting development techniques having security in mind. Therefore, metrics that evaluate code quality are needed when defining a vulnerability index.

In the present work, we just define a metric in this group:

- LoC (Lines of Code).

It is known that faults tend to occur at a specific rate per LoC, even though, this is not an exact measure, it can be an indicator of the level of vulnerability in the used software. With enough information the rate can be improved. Also, different rates can be established depending on the way software was developed, i.e., if it is a wide spread commercial product, or a tailored product; if it was developed in house, or in an outsource company, etc.

4.2 Threat Metrics

This section intends to specify a set of metrics that can be used to calculate threat index, we will present several groups of metrics each related to different threat's mechanisms. This metrics mainly try to identify what is the level of attack, perimeter defenses are subject to, if this metrics show a low attack level then we can infer that internal systems have a low threat index. Some of the presented metrics are already being collected and stored by PT. Therefore, unlike what happens in vulnerability metrics, some historical information already exists, we will analyse and use it to help us defining a normal pattern, and based on it set the threshold intervals of our model. Other metrics are product of our research work and we believe they will enrich the index's quality.

4.2.1 SPAM

SPAM is a growing threat, it is responsible, not only, for network bandwidth misuse, but is also frequently used to send phishing messages. Therefore the level of SPAM is an indicator of the overall threat level that information systems face.

This metrics' group intends to give insights about SPAM activity, even though, that its detectors are subject to false positive and false negatives; these are still being useful metrics once they can help to infere the malicious activity level.

For this group we considered the following metrics:

- messages received;
- spam detected by filtering mechanisms;
- spam detected in accepted mail.

The first metric gives us the number of mail messages classified as SPAM by anti-spam system; the second represents the number of messages that have not been classified as SPAM by anti-spam system, but subsequently came to be classified as SPAM by other mechanisms (e.g., users).

4.2.2 Virus

Nowadays virus are one of the most prevalent threats to information systems. No organization can live without an anti-virus systems to keep their systems free (as much as possible) of this scourge. The amount of virus detected is an indicator of the level of malicious activity in progress. For this group we have considered two metrics:

- virus detected in servers;
- virus detected in desktops.

In our study, the first is the most representative, since the systems we want to evaluate are hosted in servers, but the latter is also important because of the risk of contagion it represents.

4.2.3 Web Filtering

Web filtering is used to prevent users of corporate networks, accessing sites that are known to pose security risks. Access to these sites may result in computer infection and subsequent propagation to other organization's computers. Information collected from this source it is an indicator of users awareness level (or risk behavior). For this group we considered the following metrics:

- permitted accesses;
- blocked accesses.

A low level of blocked accesses shows a tendency to adopt secure behaviors, and consequently a low level of threat through this channel, while a high level of blocked accesses show a tendency for risk behavior and increases threat level.

4.2.4 Firewall

It is one of the today's most used mechanisms of defence, against information security attacks, in organizations. Firewalls are the first line of defence against external threats, usually the access to any system is protected at least for one FW (Firewall). It can provide useful information about the level of attack an organization or system is subject to. For this group we intend to collect two metrics (both split in two sub-metrics representing internal and external connections):

- permitted connections;
- blocked connections.

Blocked connections can be either mistakes or intentional attacks, both of them representing a threat for information systems. The specialization in internal, or external will be based on origin IP ranges.

4.2.5 Security Incidents

4.2.5.1 Reported by IDS

IDSs (Intrusion Detection Systems) are another essential tool in building a perimeter defense of information systems. These tools intend to detect and deter possible intrusion attempts. Is therefore important to collect metrics from these equipments in order to improve the acuracy of infromation systems risk index. During this work, it has not been possible to collect data directly from IDSs (similarily to what was done to the FWs) but we were able to fill this gap by collecting data from service desk system, which records the incidents reported by IDSs and later confirmed by security teams, these metrics' values already have false positives stripped out, and are classified in terms of impact caused. Therefore, we were able to split the metric in four sub-metrics according to their impact:

- Incidents Impact=Top;
- Incidents Impact=High;
- Incidents Impact=Low;
- Incidents Impact=Unknown.

The number of incidents actually occurred is a measure of the (in)success of the attacks perpetrated against the organization's information systems; it will give information about the threat effective ability, to cause damage. These metrics intend to measure, that ability.

4.2.5.2 Reported by Users

This group is identical to the previous one but it is based on information reported by users instead of IDS information. The incidents are also classified in terms of caused impact, even though, they don't share the same classification model. For this group we have the following sub-metrics:

- Impact=Critical;
- Impact=Intermediate;
- Impact=Low.

Chapter 5

Experimental Work

In order to validate the practical applicability of our work, we have done some experimental/exploratory work. To accomplish it, we selected an information system from PT and gathered data that could be used to "feed" our vulnerability metrics, and data related to threat metrics. The aim is to verify if it is indeed possible to gather and compute the available information in order to get the intended metrics. In this chapter, we will describe the process of data collection and metric calculation. The difficulties found to define weights and thresholds of each metric are also discussed. To define the QoP of an information system, first of all, one needs to deeply know it. Thus, the first step in defining the security risk of a system is, wherefore, to make a system assessment in order to identify which assets compound it, i.e., how many servers are part of the system, which operating systems and which software these equipments run, which defences (firewalls, intrusion detection systems, anti-virus) are in place to protect the system.

5.1 System Assessment

The system selected to this experimental work is a usual 3-tier architecture system. Figure5.1 presents a simplified version of system architecture (due to confidenciality obligations names have been changed). The system main assets are:

PTWEB001 and PTWEB002 use a load balancing mechanism. They are identical systems with respect to hardware, software, and configuration. Thus we will only present information about their components and metrics' values once, since they will be equal for both. The main components of the system are:

- Windows 2003 SP2;
- IIS (Internet Information Server):
  - Common Files;
  - FTP Server;
  - Internet Information Services Manager;

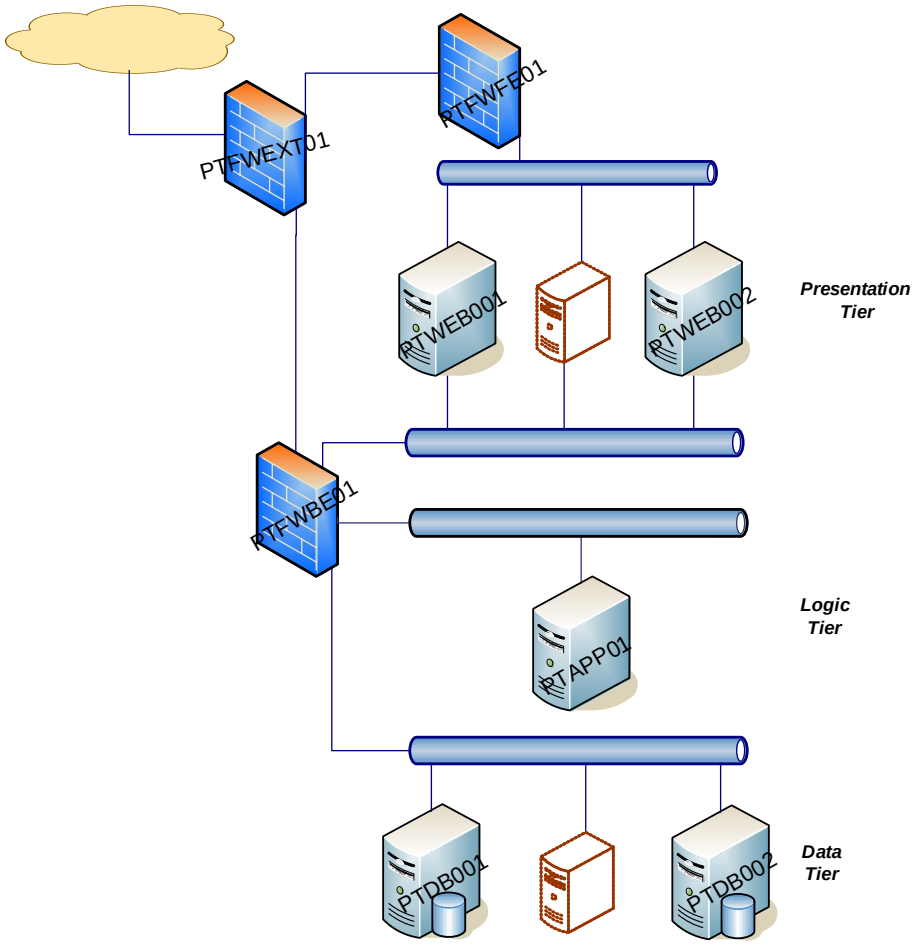


Figure 5.1: System Architecture

  - World Wide Web Server;

- Cluster Service.

PTAPP001 is a single system with no redundance

- Windows 2003 SP2;
- Application Server: Enable Network COM+ Access;
- IIS:
  - Common Files;
  - FTP Server;
  - Internet Information Services Manager;
  - SMTP Service;
- World Wide Web Server:

- Active Server Pages;
- Internet Data Connector.

PTDB001 and PTDB002 form a failover cluster with load balancing. They are identical systems with respect to hardware, software, and configuration. The cluster is used to serve several databases, each database instance will be running in one node at a time, if a node fails the other will run instances from both nodes. Thus we will only present information about their components and metrics' values for the one that is running the database instance by the time we collected the metrics. Their main components are:

- HP-UX 11.31 B;
- ORACLE 9.2.0.8.0.

5.2 Security Policy – Summary of Relevant Rules

The base set of metrics should contain those that allow to identify gaps between security policy and information systems implementation, as we stated in Section 4.1. We present here a summary of the PT policy rules that we are going to use to help us defining the weights and thresholds of the metrics.

Security policy defines strong passwords as the ones that have lower and upper case letters, digits, and special characters. Strong passwords should have at least a length of eight characteres if they belong to normal users, or twelve if they belong to privileged users. They must not have dictionary words, personal information or the user name. It also defines that password should be changed at least each 90 days.

Security policy do not define deadlines for patching installation. It just states that patches released by manufacturers should be properly applied, especially the patches related to security, with the exception of causing system malfunction. Testing and fall-back procedures must also be implemented.

5.3 Vulnerability Index – Collected Metrics

In the scope of this work, would be impossible, to cover all security aspects related to information systems. Thus, we decide to cover the elementary security groups, which we consider as being good indicators of systems' security, and sanity, namely: access control, patching, anti-virus, and exposure to known vulnerabilities. We also decided to cover three of the most relevant components of information systems today: operating system, web/application server, and database management system.

Table 5.1 shows the components and groups defined for each asset of the information system and their respective weights. In Section 5.4 we will explain the criteria used in weight definition.

Unfortunately we could not gather data to fulfill all intended metrics, therefore, when a metric's value appears with N/A (Not Available) it means, that we had planned to include that metric in our work but it was not possible to obtain data or process/compute it in time to be used in this work.

Asset		Component		Group	
Name	Weight	Name	Weight	Name	Weight
PTWEB001 & PTWEB002	1	Operating System	2	Patching	1
				Access Control	1
				Anti-virus	1
				Known Vulnerabilities	1
		Web Servers	1	Patching	1
				Access Control	1
				Known Vulnerabilities	1
PTAPP001	1	Operating System	2	Patching	1
				Access Control	1
				Anti-virus	1
				Known Vulnerabilities	1
		Application Server	1	Patching	1
				Access Control	1
				Known Vulnerabilities	1
		Bussiness Applications	1	Patching	1
				Access Control	1
				Known Vulnerabilities	1
PTDB001 & PTDB002	2	Operating System	2	Patching	1
				Access Control	1
				Known Vulnerabilities	1
		DBMS	1	Patching	1
				Access Control	1
				Known Vulnerabilities	1

Table 5.1: Assets, Components and Groups – Weights

In the subsections bellow we present the collected metrics, for each group, in each asset, and we will describe the processes used to collect those metrics, identifying the used sources, and also describing the extraction, processing and computation procedures applied.

The tables presented in this section already have columns whose values will only be explained in subsequent sections, but in order to keep all process more readable, we decided to present all information at once.

5.3.1 Presentation Tier

The presentation tier is composed by PTWEB001 and PTWEB002, as can be seen in Figure 5.1. Since as we stated before both assets are equal (hardware, software, and configuration) we will present the set of metrics and their values just once.

5.3.1.1 Operating System

Patching to compute the patching metrics shown in Table 5.2, we used two data sources:

- the information about installed patches in the systems, and their respective installation date. This set of data was provided to us by SA (System Administrator);
- Microsoft security patches lists, which we downloaded from [11].

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
	avg. days to install security patches	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
Access Control	# weak passwords found	1	-	priv. users	2	-	-
				non-priv. users	1	-	-
	% unchanged passwords > 90 days	1	-	priv. users	2	-	-
				non-priv. users	1	-	-
	# users with dangerous privileges	1	-				N/A
	# security misconfigurations detected	1	-				N/A
Anti-virus	anti-virus installed?	1	-				-
	# days old of signatures file	1	-				-
	# virus detected (day)	1	-				-
Known Vulnerabilities	# open TCP ports - iface frontend	1	-				-
	# open UDP ports - iface frontend	1	-				-
	# open TCP ports in top 10 list - iface frontend	2	-				-
	# open UDP ports in top 10 list - iface frontend	2	-				-
	# open TCP ports - iface backend	1	-				-
	# open UDP ports - iface backend	1	-				-
	# open TCP ports in top 10 list - iface backend	2	-				-
	# open UDP ports in top 10 list - iface backend	2	-				-
	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.2: Operating System Collected Metrics – Weights, Values, and Indexes (Presentation Tier)

The first gives us the effective state of the system, while the second gives a base to compare with how the system should be. Since, we received the information in a format that was not directly usable, we had to parse and extract the relevant information from the data sources, to accomplish it we developed Ruby [12] scripts, for which we constructed several regular expressions. Thereupon we uploaded the results into two MySQL [13] database tables that were then compared to extract the metrics' values.

Microsoft classifies vulnerability as critical, important or moderate according to its security impact, thus we decided to split our metrics in sub-metrics accordingly, which will allow us to define different weights for each sub-metric.

Access Control the collected metrics related to access control, their respective values, weights and indexes are presented in Table 5.2.

The data used to compute passwords' information was provided to us by SA. He extracted the SAM and system files (from %windir%\system32\config) with the server offline. Since Windows do not natively provide a process to gather passwords' age, SA used the free tool NetPWAge Net [14] to extract it.

To compute weak password metric, we started by extracting the hash information (from SAM and system files) using ophcrack [15] tool, then we used John the Ripper password cracker [16] to verify which passwords could be broken (since this is a windows system we could also have used ophcrack to do this task, but we decided to use the same password cracker for all assets, and ophcrack could not be used for HP-UX

systems), and do not respect the policy rules for strong passwords (the compliance with the security policy was "manually" verified, but it is a process that can be easily automated).

In the context of the current project, we have run John the Ripper in a 2.4GHz processor, with 1GB memory, for two days, with default parameters in incremental mode. However, it is our understanding that in a production scenario the password cracker should run for a longer period. We believe it should run until it finishes, or until the policy defined period expires, since any password that do not resist during the defined period should be considered weak.

To compute the metric of password's age, we just compared password's age against the period defined in the policy (90 days), the ones that were older than the period, were accounted in the metric. Since information related to the role of each user were also available, we decided to divide each metric in two (weak password, and password's age); classifying each account as privileged or non-privileged according to their roles, and creating metrics for privileged users and another for non-privileged users. The rational behind this division is that a broken password in a privileged account represents a higher risk.

Anti-virus the values for these group of metrics were extracted directly from the anti-virus software installed in the system, and was also supplied to us by SA.

The collected metrics related to anti-virus, their respective values, weights and indexes are presented in Table 5.2.

Known Vulnerabilities the metrics related to open ports, were collected by the SA directly from the system using the netstat command. Once this system has frontend and backend interfaces, we collected information for both and treated them separately. To define the top 10 list of open ports we collected one month (November 15^{th} to December 15^{th}) data from NoAH (European Network of Affined Honeypots) [17], [18], [19].

Since it was not possible to run a vulnerability scanner against the system, we have no data for vulnerability detected metric.

The collected metrics related to known vulnerabilities, their respective values, weights and indexes are presented in Table 5.2.

5.3.1.2 Web Server

Patching once in this case, the web server was the Microsoft IIS, we used the same process that we have used for operating system, taking into account only patches related to IIS/6.0.

Table 5.3 lists the metrics collected for this group and their weights, values, and indexes.

Access Control these web servers use integrated windows authentication, which means it uses the same credentials as the operating system; therefore we use the same process, just reclassifying who are the privileged users, since a user may have a privileged role related to web server and not in the operating system and "vice-versa".

Table 5.3 lists the metrics collected for this group and their weights, values, and indexes.

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
	avg. days to install security patches	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
Access Control	# weak passwords found	1	-	priv. users	2	-	-
				non-priv. users	1	-	-
	% unchanged passwords > 90 days	1	-	priv. users	2	-	-
				non-priv. users	1	-	-
	# users with dangerous privileges	1	-				N/A
	# security misconfigurations detected	1	-				N/A
Known Vulnerabilities	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.3: Web Server Collected Metrics – Weights, Values, and Indexes

Known Vulnerabilities it was not possible to collect data to compute metrics from this group.

5.3.2 Logic Tier

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
	avg. days to install security patches	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
Access Control	# weak passwords found	1	-	priv. users	2	-	N/A
				non-priv. users	1	-	N/A
	% unchanged passwords > 90 days	1	-	priv. users	2	-	-
				non-priv. users	1	-	-
	# users with dangerous privileges	1	-				N/A
	# security misconfigurations detected	1	-				N/A
Anti-virus	anti-virus installed?	1	-				-
	# days old of signatures file	1	-				-
	# virus detected (day)	1	-				-
Known Vulnerabilities	# open TCP ports	1	-				-
	# open UDP ports	1	-				-
	# open TCP ports in top 10 list	2	-				-
	# open UDP ports in top 10 list	2	-				-
	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.4: Operating System Collected Metrics – Weights, Values, and Indexes (Logic Tier)

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
	avg. days to install security patches	1	-	critical	3	-	-
				important	2	-	-
				moderate	1	-	-
Access Control	# weak passwords found	1	-	priv. users	2	-	N/A
				non-priv. users	1	-	N/A
	% unchanged passwords > 90 days	1	-	priv. users	1	-	-
				non-priv. users	1	-	-
	# users with dangerous privileges	1	-				N/A
	# security misconfigurations detected	1	-				N/A
Known Vulnerabilities	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.5: Application Server Collected Metrics – Weights, Values, and Indexes

In this tier, we have the PTAPP001 equipment (see Figure 5.1) which is also a Windows 2003 system with an IIS. This way all process of gathering and computing data was identical to the one used in the presentation tier. Thus in this subsection we will only show Tables 5.4 and 5.5, which present collected metrics, and we will highlight the differences when justified.

5.3.2.1 Operating System

Access Control for this group, we were not able to collect metrics related to weak passwords (the files we received with information collected from PTAPP001 server were unreadable), and it was not possible to extract the information a second time.

5.3.2.2 Application Server

Table 5.5 presents the metrics collected for application server (PTAPP01) of logic tier.

5.3.2.3 Bussiness Aplications

Due to time limitations, it was not possible to collect metrics of business applications, which demand for a deeper assessment of the system in order to know all its details.

5.3.3 Data Tier

In this tier, we have the PTADB001 equipment (see Figure 5.1) which is an HP-UX 11 system running an ORACLE 9.2 database instance. Since as we stated before both assets are equal (hardware, software, and configuration) and just one is running the database instance at a time, we will present the set of metrics and their values just for that one.

The process of data gathering was not substancially different from the prior ones, so in this subsection we will only describe the distinct or particular aspects of the process. Table 5.6 shows the metrics collected for operating system, and Table 5.7 shows metrics collected for DBMS.

5.3.3.1 Operating System

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-				N/A
	avg. days to install security patches	1	-				N/A
Access Control	# weak passwords found	1	-	priv. users	-	-	-
				non-priv. users	-	-	-
	% unchanged passwords > 90 days	1	-	priv. users	-	-	-
				non-priv. users	-	-	-
	# users with dangerous privileges	1	-				N/A
	# security misconfigurations detected	1	-				N/A
Known Vulnerabilities	# open TCP ports	1	-				-
	# open TCP ports in top 10 list	2	-				-
	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.6: Operating System Collected Metrics – Weights, Values, and Indexes (Data Tier)

Access Control the process used in this group was identical to the one used for Windows systems. The SA provided us the passwords' hashes and the last change date of each password. Once, for this system we received the passwords' hashes the step of hash extraction was not needed, so we used John the Ripper, with the same options and in the same machine, to verify which passwords could be broken. To obtain password age, we calculated the number of days since the password had been changed and the rest of the process was like the one described before for Windows systems.

Anti-virus even though, Unix virus are, at least, theoretically possible, this is far from being a prevalent threat. Therefore, these servers which run a HP-UX system have no anti-virus installed, thus this group of metrics is inapplicable to this asset.

5.3.3.2 DBMS

Access Control to compute the metrics of this group, DBA (Database Administrator) provided us information from several Oracle dictionary tables. Oracle DBMS has a complex set of roles and privileges that can be granted to users. In order to obtain fine grained metrics, we defined three levels of dangerous privileges (see Table 5.8) according to its impact potencial. Then we used information extracted from Oracle tables (DBA_ROLE_PRIVS, DBA_SYS_PRIVS, sys.USER$, DBA_ROLES) to assign the danger level to each user according to the privileges he possesses, and we counted the number of users in each group to obtain the dangerous privileges metrics.

To obtain the metric about passwords' age, we used the information from ptime column of table sys.USER$ and calculated the number of days since the password has been changed, and we grouped the information

according to the danger level of each user. For this group, we were unable to collect the passwords' hashes. Therefore the vulnerability index is not as complete as it could be, but in terms of process it makes no difference, since it would be equal to the ones presented above, i.e., we would have used John the Ripper to try to break the passwords' hashes.

Group Name	Metric Name	Weight	Index	Sub-metric Name	Weight	Index	Collected Value
Patching	# security patches not installed	1	-				N/A
	avg. days to install security patches	1	-				N/A
Access	# weak passwords found	1	-				N/A
Control	% unchanged passwords > 90 days	1	-	users danger priv. (level 1)	4	-	-
				users danger priv. (level 2)	3	-	-
				users danger priv. (level 3)	2	-	-
				non priv. users	1	-	-
	# users with dangerous privileges	1	-	level 1	3	-	-
				level 2	2	-	-
				level 3	1	-	-
	# security misconfigurations detected	1	-				N/A
Known Vulnerabilities	# vulnerabilities detected (vulnerability scanner)	1	-				N/A

Note: Confidential values removed from public version.

Table 5.7: DBMS Collected Metrics – Weights, Values, and Indexes

5.4 Vulnerability Index – Thresholds and Weights

In this stage we do not have enough information to allow correlation between metrics and incidents, i.e., we have no factual support to say that one metric should have a higher weight (and how much higher it should be) than another, because, for instance, a high percentage of incidents happened in systems that had higher values in this metric. Then as stated in Chapter 3 the weight values have to be set based on expert opinion, or by trial and error using common sense and empirical knowledge. That said, the weights presented here should be seen as a starting point proposal, which can even be a bit naive, and should not be seen as well proven scientific facts. The idea is to subject those values to expert scrutiny and validation. Gradually improving their values as new data is being collected, and a more accurate correlation is being established between data and occurrences.

Therefore, in this first approach, we tried to use simple rules to define weight values:

- when we had sub-metrics with known scales, we assigned weights in accordance with that scale, for instance, if we have a set of identical events that can be classified in: high, medium, or low according to their severity, we assigned a weight of 1 to low, 2 to medium and 3 to high;
- for the upward levels we kept identical weights for all elements, except for the following cases:
  - data tier received a superior weight (see Table 5.1) because it is the place where all data is available, therefore a successful attack to this tier tend to have a stronger impact in confidentiality than a successful attack to any of the other tiers. Different impacts in availability and integrity

Danger Level	1	2	3
Possible attacks	denial of service or data breaches	denial of service	data breaches or data scrambling
Privileges	GRANT ANY% ADMINISTER DATABASE TRIGGER ALTER USER BECOME USER	DROP ANY% ADMINISTER RESOURCE MANAGER ALTER PROFILE ALTER DATABASE ALTER RESOURCE COST ALTER ROLLBACK SEGMENT ALTER SYSTEM ALTER TABLESPACE ANALYZE ANY CREATE PUBLIC SYNONYM CREATE ROLLBACK SEGMENT DELETE ANY TABLE DEQUEUE ANY QUEUE DROP PROFILE DROP PUBLIC DATABASE LINK DROP PUBLIC SYNONYM DROP ROLLBACK SEGMENT DROP TABLESPACE DROP USER ENQUEUE ANY QUEUE FORCE ANY TRANSACTION LOCK ANY TABLE MANAGE ANY QUEUE MANAGE TABLESPACE UNLIMITED TABLESPACE	CREATE ANY% ALTER ANY%' EXECUTE ANY% AUDIT ANY AUDIT SYSTEM BACKUP ANY TABLE CREATE DATABASE LINK CREATE PUBLIC DATABASE LINK CREATE ROLE CREATE USER INSERT ANY TABLE RESTRICTED SESSION UPDATE ANY TABLE

Table 5.8: Oracle Dangerous Privileges

relating to tier are more difficult to establish. So for now we will assume they will have identical impact, no matter which is the affected tier;

- operating system received a superior weight (see Table 5.1) because it is the base where all other components rely to perform some of their critical operations;
- open ports in top 10 list, also received a superior weight (see Tables 5.2, 5.4, 5.6), once the probabilities of a successful attack tend to increase with the rise in attacks, also when a port is in the top 10 list, this usually means that knowledge of some vulnerability, in the service normally assigned to that port, has become public.

Even though, in our example, we are using only integer weights, it is, obviously, not mandatory, and would not even be advisable in some cases.

In what regards to interval thresholds, in Section 3.4 we had foreseen four methods to assign their values. But, since we could not collect historical data allowing the use of statistical analysis, nor defining what a normal pattern should be, and there is no causation evidence to support our decision then the only methods to set interval threshold at this stage is also using expert knowledge or trial and error. Bellow, more than define/present absolutely reliable interval thresholds (which, as we have said, without a considerable volume of historical data and correlation of events is impossible) we will present some arguments that should be taken into account when analyzing the data that will be used for defining those thresholds. Therefore, for

some metrics, we will present complete thresholds definition, while for others, we shall just suggest the security impact a particular metric's value may have. However, this process should be seen as a first attempt, and it is merely illustrative and done just in order to generate data to demonstrate how our model is supposed to work. The experience acquired from the continuous use of metrics, will certainly give information that can be used to improve this classification, in future developments of this work.

Patching patch management is not a consensual matter, there is a general agreement on the need of patching, but different strategies can be applied depending on several factors like: organization characteristics, downtime limitations, patching costs, or the costs to recover from patch failure. HP (Hewlett-Packard), for instance, suggests in [20] that "*Ideally, your strategy should include proactive patching, reactive patching, and a separate plan for security patches" and "The frequency and timing of patch installation maintenance windows must be chosen to meet with particular system down time limitations and the need to install the new patches.*" There is no right strategy, but for sure, security patch management must have an immediate reaction strategy, due to the increased risk it poses, but without forgetting the need for testing and implement fall-back procedures in the case of malfunction due to patch installation. Therefore it is hard to define how many missing patches and for how long can one be without patching and do not increase the level of vulnerability. The absence of a single patch always poses some security risk so only if all patches are installed we can set the no security risk threshold (value 0). Since patches need some time to be tested and deployed (which usually means some period of down-time) we can say that it is not realistic to expect that the patches released in the last month are already installed. Based on that, we use the information about security patches, released by Microsoft for windows 2003 system, in the last four years, to define this group's thresholds. We calculate the statistical mode, for the number of patches released by month, and the result was two. So that will be our "magic number" to define threshold intervals for number of patches not installed. With respect to average days to install patches metric, we consider a month as a realistic period to test and set a down-time period to deploy a security patch, thus thirty days will be used to define threshold intervals. To keep this first approach to security metrics simple, we will use the same thresholds for all assets and components that collect these metrics, even though, other components may have sparser patches' releases. Table 5.9 shows the patching thresholds defined.

Index	# security patches not installed	avg. days to install security patches
0	0	0
1	]0, 2]	]0, 30]
2	]2, 4]	]30, 60]
3	]4, 6]	]60, 90]
4	]6, +∞[	]90, +∞[

Table 5.9: Patching Thresholds

Access Control as a first approach to define interval thresholds for weak passwords, we considered two hypotheses: the percentage of weak passwords in relation to total of accounts, or the absolute number of weak passwords. Even though, that in a system with a high number of accounts the probability of find a weak password is higher than in a system with few accounts, it is also true that a weak password is as dangerous in one system as in the other, no matter the total number of accounts. Therefore, we used the total number of weak passwords. Since, we understand that weak passwords are a major weakness in systems, we decided to

be very restrictive in defining weak password interval thresholds, thus each interval will have only one unit (see Table 5.10).

About passwords unchanged for more than 90 days, once, when this metric has values higher than 0 it means that system is not enforcing security policy related with password management, this metric indicates essentially the users positioning about security. Therefore, we used a percentage metric in order to infer the population behavior. Having this in mind the first obvious approach to set interval thresholds is to divide the percentage result space in four equal parts (see Table 5.10). The rational is that the higher is the population not caring about security the more vulnerable is the system.

Usually to administer a system we need two or three accounts with “dangerous” privileges. If we think about a big organization where there are a clear separation of jobs, we can think about three different users: root which have all privileges; system administrator which have the privileges needed to perform the day by day administration; and operator which performs small administrative tasks like for instance backup and which has a more restricted set of privileges. Therefore, to define our threshold intervals we propose the use of multiples of three. For other components, besides operating system the situation is similar, so in the context of this work we will use the same threshold intervals for metric "users with dangerous privileges".

Index	# weak passwords found	% unchanged passwords > 90 days	# users with dangerous privileges
0	0	0	0
1	]0, 1]	]0, 0.25]	]0, 3]
2	]1, 2]	]0.25, 0.5]	]3, 6]
3	]2, 3]	]0.5, 0.75]	]6, 9]
4	]3, +∞[	]0.75, 1]	]9, +∞[

Table 5.10: Access Control Thresholds

Anti-virus virus always poses severe risks; therefore a system without an up-to-date anti-virus has a high level of vulnerability. Anti-virus installed can only take two values, so we will map one of them with the minimum threshold value and the other with the maximum threshold value. Number of days old of signatures file has to be measured against the signature file release date.

Given the decreasing time worms and virus take to spread after a vulnerability is released an outdated signature file is a weak defense. For instance, in March 2004, *"The Witty worm incident was unique in that the worm spread very rapidly after announcement of the ISS vulnerability (a day later)"* [21]. Therefore, once again, as a first approach we decided to increase interval for each day without update (see Table 5.11).

About the virus detected per day, in reality we reach the conclusion that it is not really a meaningful metric; usually systems are kept without virus, so this metric usually will give zero. The absolute value being collected in a daily basis will make the index to increase in the day the system is infected and decrease in the day immediately after. Therefore we could not extract any useful information from such a metric. An average or a moving average for a wider period would be more informative, since systems that are prone to infections will tend to consistently present values higher than zero, and that means something about system vulnerability. In these circumstances, we decided not to use this metric in calculating the index of vulnerability.

Index	anti-virus installed?	# days old of signatures file	# virus detected (day)
0	Y	0	0
1		]0, 1]	-
2		]1, 2]	-
3		]2, 3]	-
4	N	]3, +∞[	-

Table 5.11: Anti-virus Thresholds

Known Vulnerabilities a higher number of open ports, means a higher exposure to threats. In order to reduce the exposure to threats, systems should only open the needed ports to execute their functions. But the total number of open ports will depend on their functions; a web server may be different from a database server. Therefore without further data we decided not to define threshold interval for this metric. Instead in the calculation process, we will suggest the range of indexes where, in our opinion, the metric should fit. Then we will present a maximum and a minimum index of vulnerability.

On the other hand, to define a threshold interval for number of open ports in top ten list is almost obvious, a division by the number of index values (four in our case) will do the job (see Table 5.12) . Since we only have integer values not all intervals will have the same size, so keeping the idea of worst case the decided to keep the intervals for index one and two smaller than the other ones.

Index	# open ports	# open ports in top 10 list	# vulnerabilities detected (vulnerability scan)
0	0	0	0
1	-	]0, 2]	-
2	-	]2, 4]	-
3	-	]4, 7]	-
4	-	]7, 10]	-

Table 5.12: Known Vulnerabilities Thresholds

5.5 Vulnerability Index – Calculation

As explained in Section 3.6 to calculate vulnerability index we use the Equation (3.1), in each level of the vulnerability graph (see Figure 3.3). We already present the results of this equation applied to sub-metrics and metrics in columns named Index in Tables from 5.2 to 5.7. Therefore in this section we are just going to present the index for group component, and asset levels, their respective values can be seen in Table 5.13 (since we have not defined exact intreval thresholds, in some index cells there are two values, which are the minimum and maximum index taking in consideration the threshold range defined in some of their metrics). Finaly, just to exemplify the use of the Equation (3.1), we present the vulnerability index calculations,

Asset			Component			Group		
Name	Weight	Index	Name	Weight	Index	Name	Weight	Index
PTWEB001 & PTWEB002	1	a; b	Operating System	2	-	Patching	1	-
						Access Control	1	-
						Anti-virus	1	-
						Known Vulnerabilities	1	-
			Web Servers	1	-	Patching	1	-
						Access Control	1	-
						Known Vulnerabilities	1	-
PTAPP001	1	c; d	Operating System	2	-	Patching	1	-
						Access Control	1	-
						Anti-virus	1	-
						Known Vulnerabilities	1	-
			Application Server	1	-	Patching	1	-
						Access Control	1	-
						Known Vulnerabilities	1	-
			Bussiness Applications	1	-	Patching	1	-
						Access Control	1	-
						Known Vulnerabilities	1	-
PTDB001 & PTDB002	2	e	Operating System	2	-	Patching	1	-
						Access Control	1	-
						Known Vulnerabilities	1	-
			DBMS	1	-	Patching	1	-
						Access Control	1	-
						Known Vulnerabilities	1	-

Note: Confidential values removed from public version.

Table 5.13: Assets, Components and Groups – Weights and Indexes

$$Index\,(Vulnerability) =$$
$$= \frac{weight(PTWEB) \times index(PTWEB) + weight(PTAPP) \times index(PTAPP) + weight(PTDB) \times index(PTDB)}{weight(PTWEB) + weight(PTAPP) + weight(PTDB)}$$

$$MinIndex(Vulnerability) = \frac{1 \times a + 1 \times c + 2 \times e}{4} = v_1$$

$$MaxIndex\,(Vulnerability) = \frac{1 \times b + 1 \times d + 2 \times e}{4} = v_2$$

5.6 Threat Index – Collected Metrics

Some metrics used here were already being collected by PT, mainly to meet commitments with CSIRT's (Computer Security Incident Response Team) network. Since this data was not as plentiful as we would wish, and we also found it at a high granularity level, it is impossible (for now) to compute what could be called a system threat index (as we refer in Section 3.7). Nevertheless, we were able to collect some metrics related to the data centre where our information system is housed. Thus, we defined metrics at two different levels: organization, and data centre, which will allow us to define different weights according to the level where the metrics were collected. Also this threat index will be the same for all systems housed in the same

Level		Group		Metric		Sub-metric	
Name	Weight	Name	Weight	Name	Weight	Name	Weight
Organization	1	Security		Reported by IDS	1		
		Incidents	2	Reported by Users	1	Impact=Critical	3
						Impact=Intermediate	2
						Impact=Low	1
		SPAM	1	Detected by Filtering Mechanisms	1		
				Detected in Accepted Mail	2		
		Web Filtering	1	Blocked Accesses	1		
		Virus	2	Virus Detected in Desktops	1	Alert	3
						Critical	2
						Notice	1
				Virus Detected in Servers	2		
Data Centre	2	FW	1	PTFWEXT01 Blocked Connections	2	int→int	1
						ext→int	2
				PTFWFE01 Blocked Connections	1	int→int	1
						int→ext	2
				PTFWBE01 Blocked Connections	1	int→int	1
						int→ext	2

Table 5.14: Threat Collected Metrics and Weights

data cenrtre or sharing the same defense mechanisms. The information collected in order to calculate threat index can be seen in Table 5.14.

5.6.1 Organization Level

At organization level we gathered the metrics related with SPAM, Web Filtering, Security Incidents, and Virus. All metrics except the ones in the Virus group were already being collected by PT in a monthly base, some since January 2010, and others since June 2010. Therefore, the process of gathering these metrics was simple and straightforward, but with the disadvantage of using the data as is, i.e., we have short time series, to help us define thresholds and metrics' index.

The Virus metric was collected from McAfee ePO tool which centralizes the management of anti-virus tools. There was log information available on a daily basis since October 1^{st}, and we were able to collect information related to virus detection sub-divided into desktop detections, and server detections, each of them yet sub-divided by severity level.

5.6.2 Data Centre Level

Firewall these metrics were collected from Cisco firewall logs. After analysis of available information we decide to use metrics based on messages that were related to blocked connections, since they are the ones that give us information about unauthorised connection attempts. Due to the high number of events generated, only one of the firewalls was also registering information about established connections. We believe that a percentage metric relating the blocked with the established connections will be more accurate, since its values will have slightest changes due to increase or decrease of overall communication volume. That is, if for example the number of users increases it is expected that the overall communication increases and the

blocked connections	permitted connections
%PIX-4-106023	%PIX-6-302013
%PIX-6-106015	%PIX-6-302015
%FWSM-1-106021	%FWSM-3-710003
%FWSM-2-106007	%FWSM-6-302013
%FWSM-4-106023	%FWSM-6-302015
%FWSM-4-313004	%FWSM-6-305009
%FWSM-6-106015	%FWSM-6-305011
%FWSM-4-410001	

Table 5.15: Firewall – Log Message Codes

number of blocked connections will also increase, a percentage metric will not be affected by that fact while a cardinal metric will. Therefore we will use a percentage metric for one of the firewalls and cardinal metrics for the others.

To compute firewall metrics, we used the log messages with the codes shown in Table 5.15. A complete description of these metrics can be seen in [22], [23]. This metric was collected on a daily basis from December 16^{th} to December 31^{st}.

Unfortunately, it has been impossible to collect other metrics at data centre level. An example, of valuable metrics that could fit in this group would be the ones withdrawn from IDS, but there is no IDS monitoring this system. Anyway these types of metrics may be added as soon as they become available, in order to improve the quality of model outcome.

5.7 Threat Index – Thresholds and Weights

Problems in defining weights explained in Section 5.4 are similar to problems defining weights of the metrics used to calculate threat index, so the approach used was exactly the same. Table 5.14 shows the weight assigned to each metric.

We have considered that data centre metrics are more important than organization metrics, since the first is collected near the system we are studying and the second refer to higher level. Organization metrics may include information from several data centres.

At group level, we considered the security incidents and virus more important because they count things that already forced organization to act in order to fix up the affected assets.

About metrics, we considered SPAM detected in accepted mail more important because it was able to circumvent a defence mechanism (i.e., the SPAM filtering mechanisms). We considered virus in servers with a superior weight, since they usually are better defended and they are the object of our study, while desktop virus, only matter due to the contagion risk. A superior weight was attributed to PTFWEXT01, since it is in a position that is near the outside world than the other ones.

In what concerns to sub-metrics, the weights were attributed according to impact severity for security incidents and desktop virus. For FW metrics, the ones that refer to external connections received a superior weight than the ones that refer to internal connections.

As we stated before, for threat metrics, we already had time series (or we were able to collect it) with respective values (even if they were yet small time series), so we believed that the best way to define the

thresholds for the threat metrics was using statistical methods. Therefore, as metrics had very different scales, our first approach was to standardize these variables (mean deleted and divided by the standard deviation), which leads to numeric metrics expressed in terms of standard deviation units ([24], page 109), and thus to set the same thresholds for each metric. However, as several of our metrics were quite skewness, standardization was unsuitable. After that, we tried the adjustment of some statistical distributions (as normal, poisson, or exponential distributions) to the sample metrics, in order to define the thresholds based on theoretical quantiles of such distributions. Unfortunately, the reduced sample size associated with the wide dispersion in the data made unfeasible such procedure for most metrics. In that sense and while higher sample size are unavailable, only elementary procedures could be used. Thus, we employed some simple descriptive statistics, as five-number summaries (sample minimum, lower quartile, median, upper quartile, and sample maximum) and related boxplot enable to show underlying pattern, they were carry out to define cut off points to the required intervals. Quartiles give us helpful information, since it was our understanding that near median (percentile 50) should be the level of threat that a system or organization is used to support (normal pattern) and as we move away from the center upward or downward the threat level increases or decrease accordingly. All data analysis was carried out using Statistical Package for the Social Sciences (IBM SPSS 19.0 for Windows, [25]). Beside that common sense was necessary ([26], pages ix-xi) and in some circumstances the trial and error method was used, expecting that when more data becomes available, accurate intervals could be defined.

5.7.1 Security Incidents

5.7.1.1 Reported by IDS

Table 5.16 shows the available time series for security incidents, we used the initial eleven values to calculate metric threshold intervals and the last one was used as a parameter to calculate the threat index.

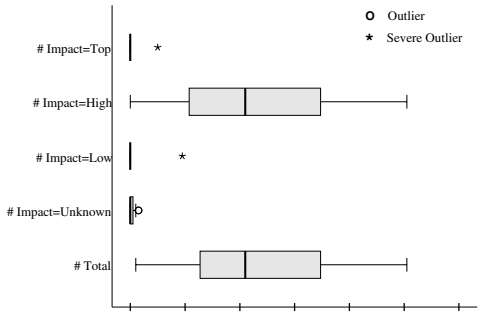
Month	# Impact=Top	# Impact=High	# Impact=Low	# Impact=Unknown	# Total
Jan	-	-	-	-	-
Feb	-	-	-	-	-
Mar	-	-	-	-	-
Apr	-	-	-	-	-
May	-	-	-	-	-
Jun	-	-	-	-	-
Jul	-	-	-	-	-
Aug	-	-	-	-	-
Sep	-	-	-	-	-
Oct	-	-	-	-	-
Nov	-	-	-	-	-
Dec	-	-	-	-	-

Note: Confidential values removed from public version.

Table 5.16: Security Incidents – Time Series

For this metric, there were sub-metrics split according to their impact, and there was a total metric. We could use the total metric, or the sub-metrics, but since they are measuring the same thing we will have no advantage in using both. Using sub-metrics and assigning correct weights to them, will tend to define a more accurate index. Nevertheless, due to the size of the sample and rareness of some events, in this case, defining

thresholds becomes, somehow, guesswork. Figure 5.2 plots parallel boxplots and Table 5.17 shows respective percentiles for this group of metrics. We can see that besides the metric with total, only the metric named "# Impact=High" give fully usable percentiles. Therefore, we are going to use the total values to derive the security incidents metric. Table 5.18 shows the defined thresholds for each metric, in accordance with what we explain above.



Note: Confidential values removed from public version

Figure 5.2: Security Incidents – Boxplots

	Percentiles		
	25	50	75
# Impact=Top	a0	b0	c0
# Impact=High	d0	e0	f0
# Impact=Low	g0	h0	i0
# Impact=Unknown	j0	k0	l0
# Total	m0	n0	o0

Note: Confidential values replaced by variables in public version.

Table 5.17: Security Incidents – Percentiles

Index	# Impact=Top	# Impact=High	# Impact=Low	# Impact=Unknown	# Total
0	0	0	0	0	0
1	]0, a0]	]0, d0]	]0, g0]	]0, j0]	]0, m0]
2	]a0, b0]	]d0, e0]	]g0, h0]	]j0, k0]	]m0, n0]
3	]b0, c0]	]e0, f0]	]h0, i0]	]k0, l0]	]n0, o0]
4	]c0,+∞[	]f0, +∞[	]i0, +∞[	]l0, +∞[	]o0, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.18: Security Incidents – Thresholds

5.7.1.2 Reported by Users

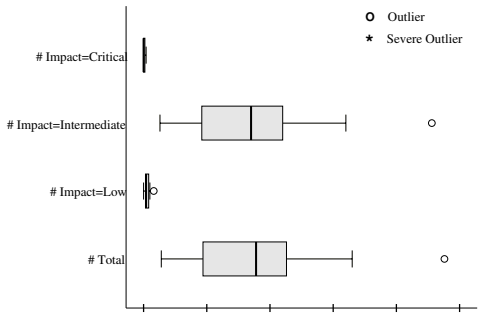
Table 5.19 shows the available time series for incident calls, we used the first eleven values to calculate metric threshold intervals and the last one was used as a parameter to calculate the threat index.

In this group, the issues to define thresholds are identical to those in the previous group. Therefore, we used the same techniques to define them. Figure 5.3 shows the boxplots for this group and Table 5.20 the corresponding percentiles. Based on them we defined the thresholds shown in Table 5.21.

Month	# Impact=Critical	# Impact=Intermediate	# Impact=Low	# Total
Jan	-	-	-	-
Feb	-	-	-	-
Mar	-	-	-	-
Apr	-	-	-	-
May	-	-	-	-
Jun	-	-	-	-
Jul	-	-	-	-
Aug	-	-	-	-
Sep	-	-	-	-
Oct	-	-	-	-
Nov	-	-	-	-
Dec	-	-	-	-

Note: Confidential values removed from public version.

Table 5.19: Incident Calls – Time Series



Note: Confidential values removed from public version

Figure 5.3: Incident Calls – Boxplots

Also, as in security incidents reported by IDS, we can use either the total values or the sub-metrics' values to derive threat index, in this case we have time series with a wide range of values, which allow us to define more complete threshold intervals. Thus, in this situation, we are going to use the partial time series (instead of the totals) essentially because they exemplify better the use of sub-metrics in our model. Nevertheless, in a production environment (where small samples also exist) some simulation must be performed over time, in order to decide whether the use of the total can produce a better index than the use of sub-metrics.

	Percentiles		
	25	50	75
# Impact=Critical	a1	b1	c1
# Impact=Intermediate	d1	e1	f1
# Impact=Low	g1	h1	i1
# Total	j1	k1	l1

Note: Confidential values replaced by variables in public version.

Table 5.20: Incident Calls – Percentiles

Index	# Impact=Critical	# Impact=Intermediate	# Impact=Low	# Total
0	0	0	0	0
1	]0, a1]	]0, d1]	]0, g1]	]0, j1]
2	]a1, b1]	]d1, e1]	]g1, h1]	]j1, k1]
3	]b1, c1]	]e1, f1]	]h1, i1]	]k1, l1]
4	]c1,+∞[	]f1, +∞[	]i1, +∞[	]l1, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.21: Incident Calls – Thresholds

5.7.2 SPAM

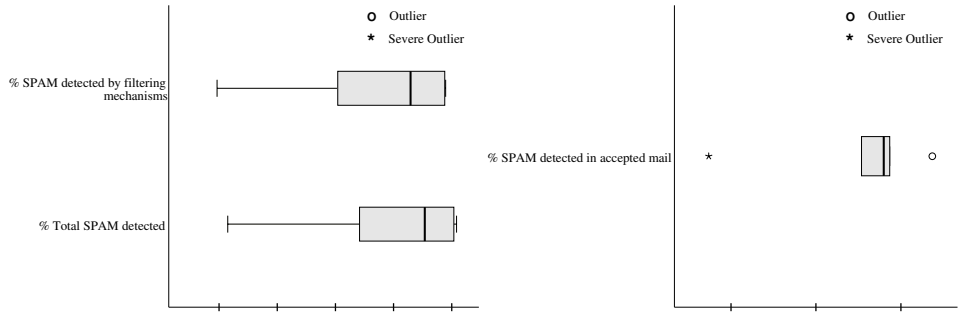
Month	# Received mail messages	# SPAM detected by filtering mechanisms	# SPAM detected in accepted mail	% SPAM detected by filtering mechanisms	% SPAM detected in accepted mail	% Total SPAM detected
Jun	-	-	-	-	-	-
Jul						
Ago	-	-	-	-	-	-
Set	-	-	-	-	-	-
Oct	-	-	-	-	-	-
Nov	-	-	-	-	-	-
Dec	-	-	-	-	-	-

Note: Confidential values removed from public version.

Table 5.22: SPAM – Time Series

For SPAM metrics we had the time series shown in Table 5.22. As explained before, the first gives the total number of messages received; the second is the number of messages blocked by the filtering mechanism; the third gives the number of SPAM messages not detected by filtering mechanisms but that afterwards were classified as SPAM; the fourth is a simple percentage of filtered messages; and the fifth is the percentage of SPAM in the mail that was not blocked by the filtering mechanism.

We had seven cases (one of them has no data available). Thus, we use the first five do calculate thresholds and the last to calculate metric index.



Note: Confidential values removed from public version

Figure 5.4: SPAM – Boxplots

Since we were able to compute percentage values (which as stated before, we believe are more accurate),

we are going to use them, instead of the cardinal metrics. And since we have two metrics (this two metrics could have been transformed in sub-metrics if we have other kind of metrics in SPAM group, i.e., metrics not related with detection) and there is no obvious reason to reject none of them, we are going to use them, instead of the total metric (as stated before, there is no gain in using total and partial metrics together).

Figure 5.4 shows the boxplots for this group and Table 5.23 the corresponding percentiles. Based on them we defined the thresholds shown in Table 5.24.

	Percentiles		
	25	50	75
% SPAM detected by filtering mechanisms	a2	b2	c2
% SPAM detected in accepted mail	d2	e2	f2
% Total SPAM detected	g2	h2	i2

Note: Confidential values replaced by variables in public version.

Table 5.23: SPAM – Percentiles

Index	% SPAM detected by filtering mechanisms	% SPAM detected in accepted mail	% Total SPAM detected
0	0	0	0
1	]0, a2]	]0, d2]	]0, g2]
2	]a2, b2]	]d2, e2]	]g2, h2]
3	]b2, c2]	]e2, f2]	]h2, i2]
4	]c2, 100]	]f2, 100]	]i2, 100]

Note: Confidential values replaced by variables in public version

Table 5.24: SPAM – Thresholds

5.7.3 Web Filtering

Month	# Permitted accesses	# Blocked accesses	% Blocked accesses
Jun	-	-	-
Jul			
Aug			
Sep	-	-	-
Oct	-	-	-
Nov	-	-	-
Dec	-	-	-

Note: Confidential values removed from public version.

Table 5.25: Web Filtering – Time Series

Table 5.25 shows the available time series about web filtering (for July and August there were no values). The number of accesses not blocked by the web filtering mechanism is in the first column, the number of attempted connections that were blocked is in the second, and the the percentage of attempted block connections is in the third.

As we stated in Subsection 4.2.3 this metric will allow us to identify the users' level of awareness about unsafe sites. It is expectable that a higher volume of accesses will correspond to a higher volume of attempted access

to blocked sites. Therefore, it is our understanding that the best time series to derive this metric will be the "% Blocked accesses".

For this metric in particular, PT has defined that it should be lower than 2%. This way, since all values gathered are lower than 2%, would make no sense to calculate the percentile values (as we have done for prior metrics), once the 2% should be what is considered as normal in prior metrics (percentile 50, or median value). Since we do not have a better way to define the remaining threshold intervals, we defined equal intervals for each one. Table 5.26 shows the thresholds defined for web filtering metric.

Index	% Blocked accesses
0	0
1	]0, 1]
2	]1, 2]
3	]2, 3]
4	]3, $+\infty$[

Table 5.26: Web Filtering – Thresholds

5.7.4 Virus

5.7.4.1 Desktop

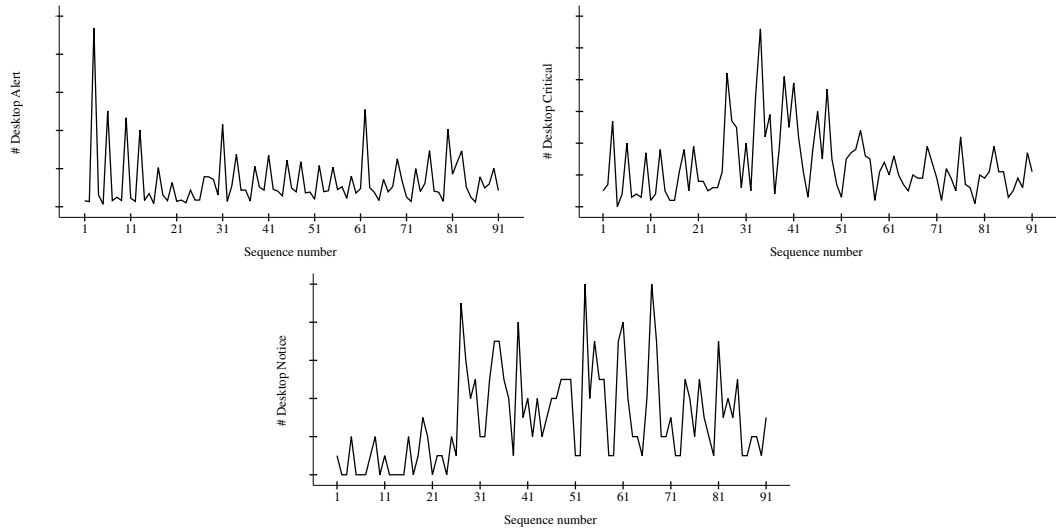
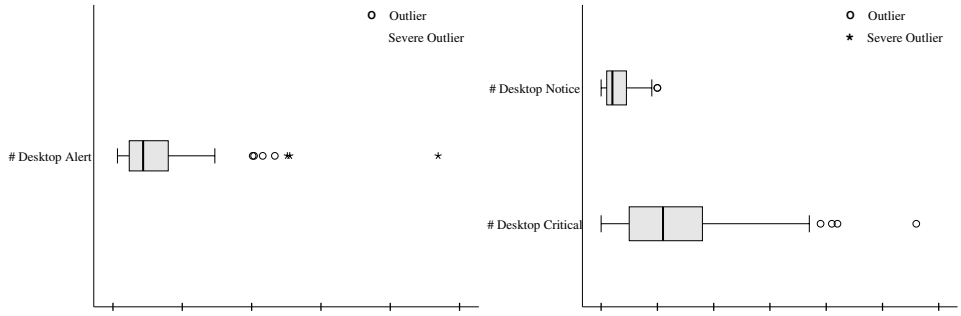


Figure 5.5: Virus Detected in Desktops – Time Series

The data available for desktop viruses, allowed us to define three sub-metrics according to the level of severity (alert, critical and notice) of the events. The graphs of Figure 5.5 show the time series used for each sub-metric. These time series have ninety two cases, similarly to what we have done in prior metrics, we used all but the last one to define thresholds and the last one to define the metric index.

Figure 5.6 shows the boxplots for each sub-metric, and Table 5.27 the corresponding percentiles. Based on them we defined the thresholds shown in Table 5.28.



Note: Confidential values removed from public version

Figure 5.6: Virus Detected in Desktops – Boxplots

	Percentiles		
	25	50	75
# Desktop Alert	a3	b3	c3
# Desktop Notice	d3	e3	f3
# Desktop Critical	g3	h3	i3

Note: Confidential values replaced by variables in public version.

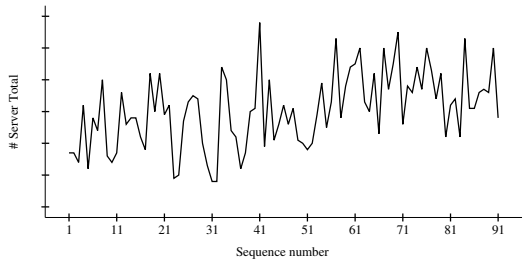
Table 5.27: Virus Detected in Desktops – Percentiles

Index	# Desktop Alert	# Desktop Notice	# Desktop Critical
0	0	0	0
1	]0, a3]	]0, d3]	]0, g3]
2	]a3, b3]	]d3, e3]	]g3, h3]
3	]b3, c3]	]e3, f3]	]h3, i3]
4	]c3, +∞[	]f3, +∞[	]i3, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.28: Virus Detected in Desktops – Thresholds

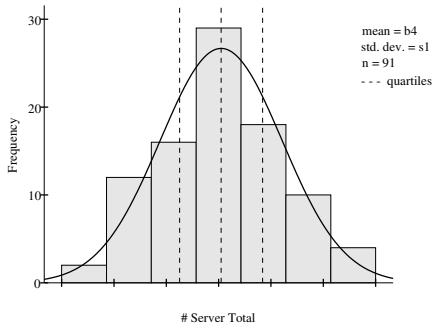
5.7.4.2 Server



Note: Confidential values removed from public version

Figure 5.7: Virus Detected in Servers – Time Series

For server virus, we also collected the same three metrics as for desktop virus. However critical and notice only have respectively one and two occurrences different from zero, also the non-zeros were all ones. Since



Note: Confidential values replaced by variables or removed from public version

Figure 5.8: Virus Detected in Servers – Histogram

there was no gains in using sub-metrics due to the difficulty in define thresholds, we used the total time series to compute Server Virus metric.

Figure 5.7 presents a graph with the time series of the total number of virus detected in servers. It also has ninety two cases, the first ninety one were used to define the thresholds and last one to compute the metric index.

Figure 5.8 shows the histogram for the total number of virus detected in servers with a normal curve. The adjustment by the curve seems fairly good and Shapiro-Wilk test was used to confirm such fit. This was the only case that we were able to adjust a statistical distribution to our data. Therefore, to define the metrics thresholds, instead of frequencies, we used the normal distribution percentiles. Figure 5.8 also shows the distribution quartiles. The equation to obtain quartiles and their values are presented in Table 5.29, and the thresholds derived from them are presented in Table 5.30.

	Percentiles		
	25	50	75
	$mean - (0.68 \times std.dev.)$	$mean$	$mean + (0.68 \times std.dev.)$
# Server Total	a4	b4	c4

Note: Confidential values replaced by variables in public version.

Table 5.29: Virus Detected in Servers – Percentiles

Index	# Server Total
0	0
1	]0, a4]
2	]a4, b4]
3	]b4, c4]
4	]c4, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.30: Virus Detected in Servers – Thresholds

5.7.5 Firewalls

We collected data from the three firewalls that protect perimeter of the information system we used in this work. The information was gathered from December 16^{th} to December 31^{st}. Similarly to what was done in prior metrics the last value was used in index metric calculation and the remaining were used to define threshold intervals.

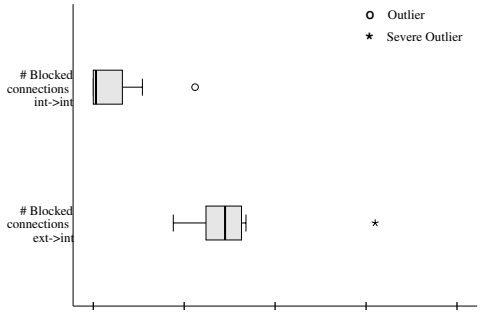
5.7.5.1 PTFWEXT01

In this firewall we have found two types of blocked connections the ones originated from external sources and the ones originated from internal sources, both have internal destination. There were no connections with external destination in this firewall. Table 5.31 shows the time series for the two types of blocked connections found in PTFWEXT01, Figure 5.9 shows their boxplots, percentiles and thresholds are presented respectively in Table 5.32 and Table 5.33.

	# Blocked connections		
Date	ext→int	int→int	int→ext
12/16/2010	-	-	
12/17/2010	-	-	
12/18/2010	-	-	
12/19/2010	-	-	
12/20/2010	-	-	
12/21/2010	-	-	
12/22/2010	-	-	
12/23/2010	-	-	
12/24/2010	-	-	
12/25/2010	-	-	
12/26/2010	-	-	
12/27/2010	-	-	
12/28/2010	-	-	
12/29/2010	-	-	
12/30/2010	-	-	
12/31/2010	-	-	

Note: Confidential values removed from public version.

Table 5.31: Firewall PTFWEXT01 – Time Series



Note: Confidential values removed from public version

Figure 5.9: Firewall PTFWEXT01 – Boxplot

	Percentiles		
	25	50	75
# Blocked connections ext→int	a5	b5	c5
# Blocked connections int→int	d5	e5	f5

Note: Confidential values replaced by variables in public version.

Table 5.32: Firewall PTFWEXT01 – Percentiles

Index	# Blocked connections ext→int	# Blocked connections int→int
0	0	0
1	]0, a5]	]0, d5]
2	]a5, b5]	]d5, e5]
3	]b5, c5]	]e5, f5]
4	]c5, +∞[	]f5, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.33: Firewall PTFWEXT01 – Thresholds

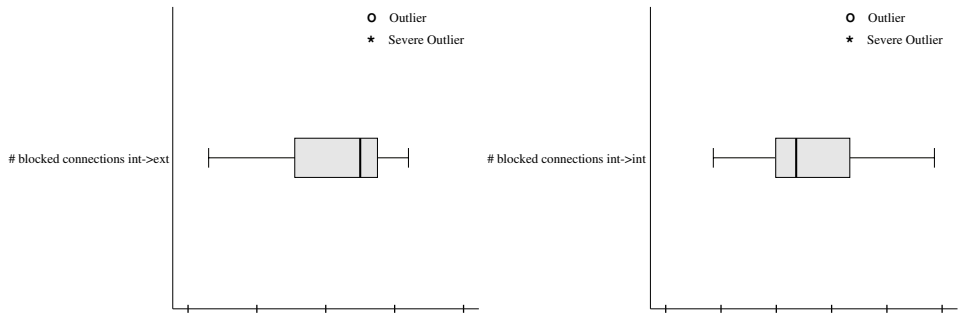
5.7.5.2 PTFWFE01

In this firewall we have found two types of blocked connections both originated from internal sources, but with destination internal and external. As expected there were no connections from external sources in this firewall. Table 5.34 shows the time series for the two types of blocked connections found in PTFWEXT01, Figure 5.10 shows their boxplots, percentiles and thresholds are presented respectively in Tables 5.35 and 5.36.

	# Blocked connections		
Date	ext→int	int→int	int→ext
12/16/2010		-	-
12/17/2010		-	-
12/18/2010		-	-
12/19/2010		-	-
12/20/2010		-	-
12/21/2010		-	-
12/22/2010		-	-
12/23/2010		-	-
12/24/2010		-	-
12/25/2010		-	-
12/26/2010		-	-
12/27/2010		-	-
12/28/2010		-	-
12/29/2010		-	-
12/30/2010		-	-
12/31/2010		-	-

Note: Confidential values removed from public version.

Table 5.34: Firewall PTFWFE01 – Time Series



Note: Confidential values removed from public version

Figure 5.10: Firewall PTFWFE01 – Boxplot

	Percentiles		
	25	50	75
# Blocked connections int→ext	a6	b6	c6
# Blocked connections int→int	d6	e6	f6

Note: Confidential values replaced by variables in public version.

Table 5.35: Firewall PTFWFE01 – Percentiles

Index	# Blocked connections ext→int	# Blocked connections int→int
0	0	0
1	]0, a6]	]0, d6]
2	]a6, b6]	]d6, e6]
3	]b6, c6]	]e6, f6]
4	]c6, +∞[	]f6, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.36: Firewall PTFWFE01 – Thresholds

5.7.5.3 PTFWBE01

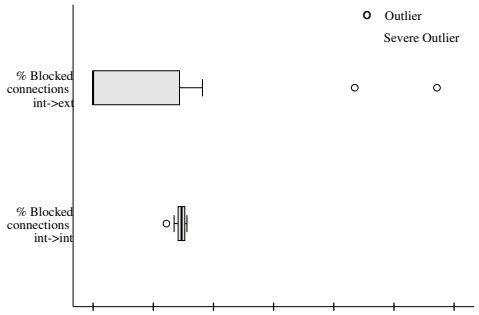
Also in this firewall we have found two types of blocked connections both originated from internal sources, but with destination internal and external. As expected, there were no connections from external sources in this firewall. Table 5.37shows the time series for the two types of blocked connections found in PTFWEXT01, Figure 5.11 shows their boxplots, percentiles and thresholds are presented respectively in Tables 5.38 and 5.39.

This firewall was the only one among the three studied, that was logging messages related to established connections, so in this case we could create time series with percentage values, which as stated before, are more immune to overall volume increases. Therefore, we used percentage values to compute this sub-metric indexes and to define the respective threshold intervals.

	# Allowed connections		# Blocked connections			# Blocked connections	
Date	int→int	int→ext	exti→nt	int→int	int→ext	% int→int	% int→ext
12/16/2010	-	-		-	-	-	-
12/17/2010	-	-		-	-	-	-
12/18/2010	-	-		-	-	-	-
12/19/2010	-	-		-	-	-	-
12/20/2010	-	-		-	-	-	-
12/21/2010	-	-		-	-	-	-
12/22/2010	-	-		-	-	-	-
12/23/2010	-	-		-	-	-	-
12/24/2010	-	-		-	-	-	-
12/25/2010	-	-		-	-	-	-
12/26/2010	-	-		-	-	-	-
12/27/2010	-	-		-	-	-	-
12/28/2010	-	-		-	-	-	-
12/29/2010	-	-		-	-	-	-
12/30/2010	-	-		-	-	-	-
12/31/2010	-	-		-	-	-	-

Note: Confidential values removed from public version.

Table 5.37: Firewall PTFWBE01 – Time Series



Note: Confidential values removed from public version

Figure 5.11: Firewall PTFWBE01 – Boxplot

	Percentiles		
	25	50	75
% blocked connections int→ext	a7	b7	c7
% blocked connections int→int	d7	e7	f7

Note: Confidential values replaced by variables in public version.

Table 5.38: Firewall PTFWBE01 – Percentiles

Index	% Blocked connections int→ext	% Blocked connections int→int
0	0	0
1	]0, a7]	]0, d7]
2	]a7, b7]	]d7, e7]
3	]b7, c7]	]e7, f7]
4	]c7, +∞[	]f7, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.39: Firewall PTFWFE01 – Thresholds

In Table 5.40 and 5.41 we present the percentiles and thresholds with cardinal values, representing the same information as the percentage values used in the calculation of PTFWBE01 threat index. Even though that, as stated before, we prefer percentage metrics, in this particular case, we can observe that calculating the threat index at December 31^{st}, the value is the same if we use percentage values or cardinal values, i.e., the threat index value for sub-metric int→int is two either with percentage, or cardinal values.

	Percentiles		
	25	50	75
# Blocked connections int→ext	a8	b8	c8
# Blocked connections int→int	d8	e8	f8

Note: Confidential values replaced by variables in public version.

Table 5.40: Firewall PTFWBE01 – Percentiles (cardinal values)

Index	# Blocked connections int→ext	# Blocked connections int→int
0	0	0
1	]0, a8]	]0, d8]
2	]a8, b8]	]d8, e8]
3	]b8, c8]	]e8, f8]
4	]c8, +∞[	]f8, +∞[

Note: Confidential values replaced by variables in public version.

Table 5.41: Firewall PTFWFE01 – Thresholds (cardinal values)

5.8 Threat Index – Calculation

Sub-metric	Value	Index	Weight
# Security incidents reported by users Impact=Critical	-	-	3
# Security incidents reported by users Impact=Intermediate	-	-	2
# Security incidents reported by users Impact=Low	-	-	1
# Virus detected in desktops - Alert	-	-	3
# Virus detected in desktops - Critical	-	-	2
# Virus detected in desktops - Notice	-	-	1
# Blocked connections int→int (PTFWEXT01)	-	-	1
# Blocked connections ext→int (PTFWEXT01)	-	-	2
# Blocked connections int→int (PTFWFE01)	-	-	1
# Blocked connections int→ext (PTFWFE01)	-	-	2
% Blocked connections int→int (PTFWBE01)	-	-	1
% Blocked connections int→ext (PTFWBE01)	-	-	2

Note: Confidential values removed from public version.

Table 5.42: Sub-metrics – Calculation Parameters

Metric	Value	Index	Weight
# Total security incidents reported by IDS	-	-	1
Security incidents reported by users	C	-	1
% Spam detected by filtering mechanisms	-	-	1
% Spam detected in accepted mail	-	-	2
% Web Filtering Blocked accesses	-	-	1
Virus detected in desktops	C	-	1
# Virus detected in servers - Total	-	-	2
PTFWEXT01 Blocked connections	C	-	2
PTFWFE01 Blocked connections	C	-	1
PTFWBE01 Blocked connections	C	-	1

C - Compute from sub-metrics

Note: Confidential values removed from public version.

Table 5.43: Metrics – Calculation Parameters

Group	Index	Weight
Security Incidents	-	2
SPAM	-	1
Web Filtering	-	1
Virus	-	2
FW	-	1

Table 5.44: Groups – Calculation Parameters

Level	Index	Weight
Organization	k	1
Data Centre	z	2

Table 5.45: Levels – Calculation Parameters

As we can see from previous section, metrics have been collected with different frequencies, some are in a monthly basis, others in a daily basis and there is no reason that prevent us to have other frequencies in future

metrics. Therefore, threat index value must be refreshed at the same rate as the metric that have a lower gathering interval, i.e., if the most frequent metric is a daily basis metric, then threat index value must be refreshed in a daily basis. Each metric's value will be valid until a new one exists, for instance a metric that is collected once a month will maintain its value during that month.

Next we are going to present a simulation of threat index calculation as if it has been computed in December 31^{st}, which is the last day from which we have collected data.

As we explained before, we have left the last value of each metric out of threshold definition, in order to use them in the index calculation, these are the values that we are going to use now.

The first step consisted in compute threat index of each metric or sub-metric, based on the defined threshold intervals. Table 5.42 presents a summary of gathered sub-metrics, their collected values, the index corresponding to each metric's value, and the assigned weights. Based on that, we computed the corresponding metrics according to the Equation (3.1), for metrics that were not split in sub-metrics, we computed their indexes according to defined threshold intervals (likewise to what was done to sub-metrics), Table 5.43 summarizes calculation parameters of metrics. Then the index of each node, of threat branch, of the tree of our model should be computed (see Figure 3.4), also according to the Equation (3.1). Obtained indexes, and group weights are presented in Table 5.44. Similarly, Table 5.45 shows the values obtained for level nodes and their weights. The last step of threat index calculation is shown in the equation:

$$index\,(Threat) =$$

$$= \frac{weight\,(Organization) \times index\,(Organization) + weight\,(DataCentre) \times index\,(DataCentre)}{weight\,(Organization) + weight\,(DataCentre)} =$$

$$= \frac{1 \times k + 2 \times z}{1 + 2} = t$$

5.9 Risk Index and Analysis of Results

As explained in Section 3.8, the Risk index will be a simple product of the Threat index by the Vulnerability index. Since, we do not calculate exact indexes for some vulnerability metrics, instead we suggested two possible values, therefore our risk index, also will have two possible values:

$$MinIndex(Risk) \quad = \quad index(Vulnerability) \times index(Threat) = v_1 \times t.$$

$$MaxIndex(Risk) \quad = \quad index(Vulnerability) \times index(Threat) = v_2 \times t.$$

Confidential. Removed from public version.

Chapter 6

Future Work

Our vision of this theme and what its implementation should be, do not ends with the end of this project. We think this is a work in progress and some other functionalities and research should be conducted in order to achieve an effective QoP measurement process.

In our work we identified the need to collect metrics related with access control misconfigurations, unfortunately we were not able to collect and to compute them. Nevertheless, it is our understanding that, one of the first improvements to this work should be including these metric values. We believe a good starting point to realize this goal is a practical evaluation of Baaz system [10], which was created having in mind to support SAs in the task of access control configuration, but we are convinced that, it can be used as a source for misconfiguration metric.

Some works have been developed ([27] is an example) with the objective of determining the vulnerability of a software component. Based on their publicly known vulnerabilities, extracted from CVE (Common vulnerabilities and Exposure), and CVSS (Common Vulnerability Scoring System), or identical sources, these works derive a software vulnerability rates. Despite, we do not further explore this path in our current work, we are convinced that it is possible to use such a system in replacement of the results of a vulnerability scanner. In fact a vulnerability scanner will look for these same vulnerabilities, and will only return (theoretically) the ones that have not been patched, which tend to be a more accurate result. Nevertheless, mainly due to the issues posed by running a vulnerability scanner in a production environment, replace it with a value that indicates the propensity of a piece of software to be vulnerable, can represent a good compromise.

The practical component of this project was made totally offline, i.e, the data was extracted from their origin systems and was processed with ad-hoc mechanisms to extract the defined metrics. In order to be put in production, such a project needs to automate the gathering and the computational processes. In PT, some of Pulso QoS infra-structures can be used, namely to extract data. The main concern would be the reinforcement of confidentiality mechanisms, since security metrics demands for higher standards, due to the sensitivity of the information gathered.

The deployment of a project like this must take into account a presentation layer. However, Pulso already implements the necessary mechanisms (dashboards, alarm management, graph engine, etc) to present results. This way, the process of add the security related information will be straight forward.

In the future, new metrics which will help to improve the indexes' accuracy, or which add new security perspectives, must be added. Our model was created having that in mind, therefore as soon as those metrics are collected, they can be readily incorporated.

An evaluation process should be defined in order to determine the effectiveness of the model outcome and identify possible improvements, i.e., compare the outputs of the model with real live facts and identify, for instance weights that are to high or to low and should be tuned, or metrics that are not being as meaningful as it was initially supposed.

We will need to think about how the relation with other systems may impact the security of a system. This subject is not treated in this work, but it is well known that trust relations are sometimes established among systems (database links among different DBMSs, remote trust logins among operating systems). Do these facts impact systems' security? We believe they do. Some of this impact it is already reflected in the metrics we collected, but to know their full impact a deeper study need to be done.

We wonder if we could use this model and some intrusion cost information to calculate a cost associated with the risk. So it is our believe, this is a path which should be explored in future.

Chapter 7

Conclusions

The experimental work developed, in the context of this work, shows the difficulty of collecting security metrics, even when we have a small set of metrics, like the one used in the current work, it can be hard and take too long to gather. Thus, this should be an incremental work; one can start with a set of possible metrics and grow from there, by adding new metrics to model as soon as they become available.

We used metrics that can be obtained automatically from systems, but we still need to deeply rely on human judgment to define good weights and thresholds, which means that the final outcome of our model is yet highly dependent of human factors, and the result depends, more than we wish, of the quality of these factors. However, we believe that with enough historical information, collected from several systems and the use of techniques like sensitivity analysis, more meaningful values can be obtained.

When dealing with security metrics, extra care must be put into data gathering. Some of the metrics proposed in this work are related to sensitive information, to extract this information from their original systems pose some extra security risks, which means that by measuring we may be increasing the security risk of the information. Therefore in a productive environment, metrics should, whenever possible, be computed locally and only the results should be transmitted to be used by the QoP process. For some metrics used in our work may even be advisable not to collect them. For instance, during the development of this work, we extracted password files from the systems and verified passwords' weakness, the information was extracted from the original system and then copied to offline equipment where the password cracker was installed. After metrics have been computed, the equipment disks and the support used to transport the data were carefully erased in order to avoid posterior retrieval (either by undeleting the files, or by using techniques of data studying from magnetic fields on the disk platter surface). In a production scenario, it is not practical to use these processes due to the expected volume of monitored systems which will bring difficulties in controlling the quality/security of the automated process. To overcome the need of extracting data from the system, the cracker could be installed in the equipment were the data resides and the metric would be computed locally, but also this solution will bring: some security risks, issues related with support contracts, and deployment difficulties (for instance in windows, tools to dump password hashes are usually signalled as viruses by the anti-virus system, and some others can only access the password files if the system is offline, neither one is acceptable in a production environment). Therefore in our opinion password weakness validation is extremely useful in security auditing because, in spite of the existence of a policy defining password rules, sometimes the policy is not enforced and the rules are not accomplished, but might not be advisable to use it regularly as

source of a security metrics process.

Beyond all this, even the computed values may be used maliciously, therefore they should only be made available to authorized personal and their transmission from the origin system to the security metric management system has to be made in a secure way.

Besides the existence of security policies, which define rules that systems must meet to become more secure. It is necessary, within organizations, to be given visibility to the lack of adoption of security rules. The existence of a system of metrics, as the one defined in this study, which identifies the level of risk of each system, and highlights the system's flaws is an important step in raising the security awareness of IT professionals, and users .

With this work we make no claims to have found the method or the best way to calculate the security index of an information system. We believe this is a work in progress, which present mechanisms that can be the basis for the practical implementation of metrics' systems. We would be pleased if this work was the basis for new research, or was used by organizations as a starting point (that need to be improved through the usage experience) to calculate the security index of their systems.

Bibliography

[1] A.J.A. Wang. Information security models and metrics. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 178–184. ACM, 2005. ISBN 1595930590. 1

[2] Andrew Jaquith. *Security metrics: replacing fear, uncertainty, and doubt*. Addison-Wesley Professional, 1 edition, April 2007. 1, 1, 4.1.2.1

[3] R.A. Caralli, J.F. Stevens, L.R. Young, W.R. Wilson, and CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. *Introducing octave allegro: Improving the information security risk assessment process*. Citeseer, 2007. 2.1

[4] Octave. `http://www.cert.org/octave/methodintro.html`. 2.1

[5] M. Howard, J. Pincus, and J. Wing. Measuring relative attack surfaces. *Computer Security in the 21st Century*, pages 109–137, 2005. 2.3

[6] Paulo Veríssimo and Luís Rodrigues. *Distributed systems for system architects*. Advances in distributed computing and middleware. Kluwer Academic, 2001. ISBN 9780792372660. URL `http://books.google.pt/books?id=oOzwLX1_bpkC`. 3.3

[7] J. Alegria, R. Ramalho, and T. Carvalho. Uma experiencia open-source para "tomar o pulso" e "ter o pulso" sobre a funcao de sistemas e tecnologias de informacao. 2004. 3.3

[8] Bob Martin, Mason Brown, Alan Paller, and Dennis Kirby. *2010 CWE/SANS Top 25 Most Dangerous Software Errors*. MITRE,SANS, 2010. URL `http://cwe.mitre.org/top25/archive/2010/2010_cwe_sans_top25.pdf`. 4.1.1.2, 4.1.1.3

[9] Thomas J. Parenty. *Digital Defense: What You Should Know about Protecting Your Company's Assets*. Harvard Business Press, 1st edition, 2003. ISBN 1578517796. 4.1.2.2, 4.1.2.4

[10] Tathagata Das, Ranjita Bhagwan, and Prasad Naldurg. Baaz: A system for detecting access control misconfigurations. In *USENIX Security Symposium*, pages 161–176, 2010. 4.1.2.2, 6

[11] Microsoft security patches. `http://www.microsoft.com/technet/security/current.aspx`. 5.3.1.1

[12] Ruby. `http://www.ruby-lang.org/en/`. 5.3.1.1

[13] Mysql. `http://www.mysql.com/`. 5.3.1.1

[14] Netpwage. `http://www.optimumx.com/`. 5.3.1.1

[15] Ophcrack. http://ophcrack.sourceforge.net/. 5.3.1.1

[16] John the ripper. http://www.openwall.com/john. 5.3.1.1

[17] Noah. http://www.fp6-noah.org/. 5.3.1.1

[18] J. Kohlrausch. Experiences with the NoAH Honeynet Testbed to Detect new Internet Worms. In *IT Security Incident Management and IT Forensics, 2009. IMF'09. Fifth International Conference on*, pages 13–26. IEEE, 2009. 5.3.1.1

[19] G. Kontaxis, I. Polakis, S. Antonatos, and E.P. Markatos. Experiences and observations from the NoAH infrastructure. 5.3.1.1

[20] HP. Patch management user guide for hp-ux 11.x systems. http://bizsupport1.austin.hp.com/bc/docs/support/SupportManual/c01919407/c01919407.pdf. 5.4

[21] Witty (computer worm). http://en.wikipedia.org/wiki/Witty_(computer_worm). 5.4

[22] *Catalyst 6500 Series Switch and Cisco 7600 Series Router Firewall Services Module - System Log Messages Guide*. Cisco Systems, Inc., 2010. URL http://www.cisco.com/en/US/docs/security/fwsm/fwsm41/system/message/fwmlog41.pdf. 5.6.2

[23] *Cisco PIX Firewall System Log Messages*. Cisco Systems, Inc., 2003. URL http://www.cisco.com/en/US/docs/security/pix/pix63/system/message/63_ermsg.pdf. 5.6.2

[24] D. de Vaus. *Analyzing social science data*. SAGE, 2002. ISBN 9780761959380. 5.7

[25] IBM SPSS. http://www.spss.com/. 5.7

[26] C. Chatfield. *Problem Solving: a Statistician's Guide*. Chapman and Hall/CRC, Boca Raton, Florida, second edition, 1995. 5.7

[27] J.A. Wang, H. Wang, M. Guo, and M. Xia. Security metrics for software systems. In *Proceedings of the 47th Annual Southeast Regional Conference*, pages 1–6. ACM, 2009. 6

[28] B. Schneier. *Secrets and lies: digital security in a networked world*. Wiley computer publishing. John Wiley, 2000. ISBN 9780471253112.

[29] *ISO/IEC 27004 Information Technology - Security Techniques - Information Security Management - Measurement*. ISO/IEC, 2009.

[30] P.G. LEON and A. SAXENA. An approach to quantitatively measure Information security. 2010.

[31] I. Tashi et al. Efficient security measurements and metrics for risk assessment. In *The Third International Conference on Internet Monitoring and Protection*, pages 131–138. IEEE, 2008.

[32] DS Bhilare, AK Ramani, and S. Tanwani. Information Protection in Academic Campuses: A Scalable Framework. *Journal of Computer Science*, 4(10):864–870, 2008. ISSN 1549-3636.

[33] A. Arora, D. Hall, CA Piato, D. Ramsey, and R. Telang. Measuring the risk-based value of IT security solutions. *IT professional*, 6(6):35–42, 2004. ISSN 1520-9202.

[34] M. Narang and M. Mehrotra. Security Issue–A Metrics Perspective. *International Journal of Information Technology*, 2(2):567–571, 2010.

[35] S. Myagmar, A.J. Lee, and W. Yurcik. Threat modeling as a basis for security requirements. In *Symposium on Requirements Engineering for Information Security (SREIS)*. Citeseer, 2005.

[36] Elizabeth Chew, Marianne Swanson, Kevin Stine, Nadya Bartol, Anthony Brown, and Will Robinson. *Performance Measurement Guide for Information Security*. NIST, rev. 1 edition, 2008. URL `http://csrc.nist.gov/publications/nistpubs/800-55-Rev1/SP800-55-rev1.pdf`.